Paper

An Online Stream Monitoring Algorithm for Fraud Detection in the Transport of Goods

Paweł M. Białoń

National Institute of Telecommunications, Warsaw, Poland

https://doi.org/10.26636/jtit.2020.146620

Abstract— The process of monitoring vehicles used in road transports plays an important role in detecting fraud committed by drivers. Algorithm designers face a number of challenges, including large number of vehicles monitored, demands related to online calculations, and ability to easily explain fraud alarms triggered to supervisors who make final decisions about actions to be taken. In this paper, we propose rather general, lightweight stream, online heuristics. The vehicle's position is periodically controlled by a GNSS device. The algorithm detects potential illegal activities along the route between the origin and the destination. Anomalies in the vehicle's trajectory are detected, based on a multi-resolution analysis of the economy of routes. The economy metric is easily understood and verifiable by controllers. The solution is also capable of identifying clearly suspicious trajectories that popular geofencing approaches would overlook. The scale on which the solution may be adopted is obtained thanks to the stream – like nature of the algorithm: essentially, the resources used do not increase along with the size of the input stream (the number of GNSS frames generated for the vehicle). An experiment illustrating the algorithm's viability is presented as well.

Keywords—fraud detection, stream processing, transport monitoring.

1. Introduction and Related Work

The process of monitoring road transport operations [1] is usually understood as using a global navigation satellite system (GNSS) – in more cases GPS – to control the position of vehicles. A vehicle is equipped with an on-board unit (OBU) which identifies its GPS position and sends it, periodically, over the Internet, to a host computer system. An operator checks the position of the vehicle in order to control compliance with predefined routes and time schedules, to ensure the safety of the truck and its crew, and to identify any irregularities, such as extra pickups or drops.

Let us focus on the last activity, i.e. detection of fraud. Usually, commercial vehicles travel between predefined origin and destination locations, transporting goods between these points. A similar situation is dealt with in the case of taxi operations, where the passenger needs to be transported from a particular starting point to a specific destination. In all such situations, there is a risk that the driver may commit fraud by transporting some additional goods or passengers, on their own initiative, along with the legal load. This may require the driver to visit additional locations along his/her route, to illegally drop or pickup additional goods. Drivers may also take care of their private business while working. The methods used to detect such cases of frauds are based on various concepts.

Such methods may be online- or offline-based. In an online analysis, a suspicious behavior of the vehicle is detected by the system quickly after or even before the forbidden behavior takes place. For example, the system detects that a truck is driving in the wrong direction, or that it takes a lengthy break between the loading/dropping locations. An efficient online detection system may be used for triggering an alarm informing monitoring center staff about such events. They, in turn, may contact the driver or send other employees to check on the suspicious vehicle. An offline analysis consists in analyzing journeys after these have been completed (one-by-one or as a set), in order to detect potential fraud.

The detection methods may be used on different scales. The number of vehicles monitored may grow rapidly in large companies. Monitoring systems may also be operated by governments, countrywide, and may cover considerable number of vehicles. For example, road transports of selected goods have been monitored in Poland since 2017, pursuant to the "Act on the goods transport monitoring system" dated 9 March 2017. The main goal of the solution is to control the movement of goods subject to the imposition of excise duty predominantly fuels. The key objective is to prevent unauthorized or unofficial loading or unloading operations that would result in avoiding the payment of taxes. Pursuant to the aforementioned Act, any operations involving transportation of specific goods must be preceded by a suitable notification submitted via a government-operated online system, known as SENT [2]. The notification contains: a CN (or the Polish PKU) classification code of the goods concerned, their quantity, address of loading and unloading locations, details of contractors involved, date on which the transport operation begins, vehicle details (mainly the license plate number) and owner's details. As of the end of 2018, vehicles transporting goods monitored with the use of the SENT system must also be monitored by GNSS. Therefore, a vehicle must be equipped with a mobile phone with a suitable tracking app. Alternatively, an on-board unit (OBU) already installed in the vehicle may be used in connection with the host system to collect information required by the government. A single frame of tracking information contains primarily GNSS coordinates of the truck, GNNS time corresponding to the specific position, and identification data (of the OBU). There are two sources of information: SENT declarations and tracking information. Several authorities are allowed to control the vehicles in order to verify, whether the actual status of the vehicle matches the declarations submitted.

From the point of view of efficiency of the monitoring process, it is important how the vehicle's position is analyzed within the company. It seems that manual (visual) supervision seems to be the most common approach. The position of a vehicle is displayed on a map, along with its speed and trajectory. This information is analyzed by the operator. Interestingly, according to practitioners, may systems of this type are based on manual analysis. OBUs obtain GPS information periodically (in predefined intervals, upon a predefine distance has been covered, or of the speed has changed by a specific value). It needs to be borne in mind, however, that it is not always possible to send GPS data (to the host) online. The Internet connection may be temporarily unavailable due to switching between different base stations, the vehicle may be present in coverage holes, or the OBU may be switched off. Therefore, OBUs usually buffer GPS frames and retransmit them after the Internet connection has been regained. Interestingly, it is often the case that the current position is transmitted first, followed by the delayed frames. The current position of the vehicle is more important for the operator than its historical location. Therefore, OBUs do not preserve the order of GPS frames while sending. This indicates (though, clearly, does not determine with certainty) that the analysis performed is either manual or, even if automatic, is based on some simple logic that does not require the true order of GNSS frames. A clear drawback of the manual approach consists in its dependence on the mental and physical condition of humans performing the analysis, who are capable of supervising a small number of vehicles only. In order cope with this, new systems are usually equipped with automated solutions that are based on some simple logic. The logic uses additional information from vehicle sensors (see [3]). Load sensing capabilities are particularly useful, since they allow to easily detect anomalies in the weight of the commodity, caused by illegal drops or pickups made during the trip. Fraud evidence provided by such sensors is of the solid variety, since it is difficult to explain unplanned changes in load weight. On the other hand, equipping vehicles with specialized sensors is not always possible due to cost, compatibility and scale-related considerations.

Geofencing [4]–[6] is a good example of the automatic analysis-based approach. The area in which the vehicle may be present is defined, and each violation of the borderline triggers on alarm that calls for specific action to be taken. The permitted region is often defined as a corridor along a predefined route. The logic of geofencing is simple and an alarm may be generated based on analyzing a single frame, independently of the previous frames. The geofencing algorithm does not require a lot of memory. As explained in subsequent sections, one the main difficulties associated with geofencing consists in the need to deal with a potentially large number of routes between two points within the road network. This problem is faced unless we force the vehicle to use a particular route (which is not always viable, e.g. when transport monitoring systems are operated by governments). Some route variants can be complicated and winding, which makes them clearly uneconomical and fraud-suspicious, but still permitted by geofencing. Moreover, drivers may be required to make small detours to rest areas, service stations or by accident, when they get lost along the way or make a mistake at an intersection. Geofencing might be too sensitive for such scenarios.

Not surprisingly, various machine learning techniques have been deployed to detect anomalies and fraud in transport. Various methods are used here, including statistical approach [7], statistical methods combined with the Dempster-Shafer evidence theory [8], separation through support vector machines [9], just to name a few works. Such approaches usually compare the current route to what they have learned about former routes during the learning phase. The approaches are perfectly capable of modeling proper routes inherited from the mathematical method relied upon a while learning. However, they also suffer from several drawbacks. The first one is shared with geofencing: there are the plenty of possible routes between any two points, which are equally reasonable and are characterized by similar lengths and, therefore, costs. Nonetheless, if we compare such routes using a simple similarity measure, e.g. based on distances between trajectory points, routes that are almost equal become dissimilar. This phenomenon complicates the learning process and hinders the process of identifying a good model. The second problem consists in the verdict given by the algorithm (e.g. detection of a suspicious route) being difficult to explain to a supervisor. This is a known drawback of many learning methods, but it plays an important role in our application. Usually, the algorithm triggers a suspicious route alarm that is later taken over by an operator whose role is to either initiate verification, commerce a pursuit or cancel the alarm. In order to make the decision, it is desirable that the operator knows why the system has triggered the alarm. Another drawback of learning methods consists in the fact that the learning phase has to take place, which means that the system cannot start operating without having collected a reliable history of routes, possibly manually marked by an operator as valid or fraud prone. Such approaches may also be of the heavy-weight nature, both in terms of their conceptual complexity and computing resources required, especially during the learning phase. Selected authors are aware of these drawbacks and apply various measures to prevent them. For example, in [7], a one-class learning approach is used which simplifies the learning phase by making it human-independent. Addition of the two-class learning approach, when a human introduces negative (suspicious) examples, is possible as well. This optimizes the learning effect and also allows the learned model to follow human way of reasoning. The authors of [8] observe the phenomenon of multiplicity of equivalent routes. Thus they assess the current route not only by a usual point-distance similarity measure in reference to typical routes, but also by examining the total length of the connection. The expected distribution of the length for a given starting and ending point is calculated based on the historical files. Sadly, the authors only consider the total length of the vehicle's trajectory. They do not deal with the sections of fragments of the trajectory, potentially omitting some local deviations from the proper route. An interesting approach to online work and to the large scale of computing is proposed in [10]. As usual, a dissimilarity measure is used to compare the current trajectory with the historical ones. The potentially resource-heavy method is accelerated with special indexing of trajectories performed via the so-called local clustering and vantage trees.

In this paper, a fraud detection method based on on-line knowledge concerning development of the vehicle's trajectory is proposed. When the system detects an anomaly in the trajectory fragment observed so far, it triggers an alarm that draws the attention of the supervisor and requires them to take suitable action or to disregard the alarm. The proposed approach is based on a multi-resolution analysis of the route's economy, measured based on trajectory zigzagging (deflection from a straight line). This measure offers a multi-resolution functionality: both the shape of the entire route and the lengths of its fragments of various sizes are examined. The intention is to eliminate, from the algorithm, the many drawbacks referred to above. The approach accepts various route variants between the origin and destination locations and treats equally, as long as they exhibit similar economy metrics. It is conceptually easy and understandable for the operator assessing the alarm, who may easily assess the zigzagging of the route themselves. It is ready to run out of the box, without a lengthy and complicated learning phase, but some parameter tuning may still be necessary. The algorithm is also lightweight, facilitating its application in large-scale systems. It utilizes a para-stream character, and thus is suitable for on-line monitoring applications. Stream processing is strictly a synonym for "ideal data processing algorithm" [11]. Such an algorithm is characterized by constant memory usage per the size of input data. Thus, if the input data has the form of a stream of data, i.e. a set of GNSS frames for a given vehicle, memory usage does not reach a certain limit as the stream proceeds.

2. The Algorithm

The algorithm described in this section is used for detecting anomalies in the economy of the observed moving trajectory. Normally, a vehicle should approach its destination using a straight and short route. Whenever this requirement is not met, the route is treated as suspicious. There is a possibility (risk) that deviation from the straight line was caused by illegal activity.

The concept of the algorithm may be easily explained by comparing it with geofencing. In geofencing, we define *corridors* along the projected route; whenever the corridor is left, an alarm is triggered. The most obvious drawback of this approach is the fact that multiple potential routes exist between the origin and the destination, as explained in the example of the grid layout of roads, as depicted in Fig. 1. Routes A, B, C look equally reasonable and the number of similarly reasonable routes grows exponentially along with the increase of the grid size. Therefore, it is not obvious how a specific corridor should be defined. If we define a "grid of corridors", i.e. those existing along all road segments present in the grid, we also accept such routes as D, which is lengthy and winding, and thus clearly suspicious in terms of potentially illegal loading and unloading operations.

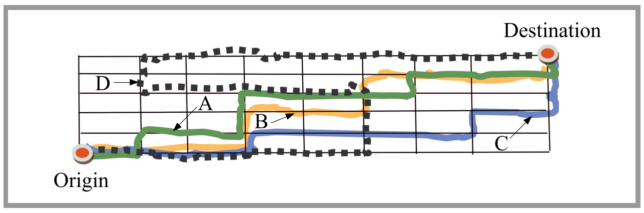

Fig. 1. Some potential routes between two points in a grid of roads. (See digital edition to find the color version).

Another drawback of corridors is that they do not allow any small deviations to rest areas, parking lots, tire shops, etc., especially if reaching these requires some maneuvers on one way roads and clover leaf interchanges.

Thus, in the proposed system, geofencing is replaced by an analysis of trajectory zigzagging. For given a trajectory T with points $x_1, x_2, \ldots, x_n$ on a plane, we define its zigzagging $Z(T)$ as:

$$Z(T) = \frac{\sum_{i=1}^{n-1} \operatorname{dist}(x_i, x_{i+1})}{\operatorname{dist}(x_n, x_1)} , \qquad (1)$$

where $\operatorname{dist}(x,y)$ denotes the Euclidean distance between x and y. Alternatively, the length of the shortest path between x and y along the existing roads could be used for dist if map information is available, as would be discussed later. Zigzagging is simply the ratio between the length of the trajectory and the distance between its end points. The main idea is to set a limit value ($Z^{\max}$) for the permitted zigzagging of vehicle trajectories. If trajectory zigzagging exceeds $Z^{\max}$, the trajectory becomes suspicious and illegal activity is taken into consideration. The rationale behind such a setting is that zigzagging routes suffer from bad economy – if the driver does need to go to illegal locations, they would not need to take such routes. Note, for example, that routes A, B, C (if represented as sufficiently dense discrete trajectories, i.e. trajectories with sufficiently large numbers of points) are characterized by a zigzagging measure of approx. the square root of 2. Route D has a larger zigzagging measure. This example shows that should $Z^{\max}$ be set at say 2 or a little more, in order to allow routing in a more complicated road grids.

Assessment of simple zigzagging measure values of the entire trajectory between the origin and destination points does not eliminate the potential of falsely evaluating the correctness of a given route based on zigzagging. It needs to be noted that if we take any fragment of the trajectory into consideration, it also should be economical in itself, i.e. should be characterized by a limited zigzagging measure. To cope with this, a *multiresolution analysis* of the route traveled so far, at a given point of the journey, should be performed. The zigzagging measure of the entire trajectory recorded so far, of a fragment of the trajectory ending at the current location, or even of a smaller fragment of the trajectory ending at the current location may be analyzed.

Let us assume that the vehicle's route is given by the stream of points x_0, x_1, x_2, …, with x_0 being its origin. The proposed stream algorithm, invoked upon receiving frame x_i, is outlined as Algorithm 1 (variable *therewasanalarm* is initialized to false).

Algorithm 1. Response to a new trajectory point x_i

```
1: If therewasanalarm Then
2: Return
3: End if
4: Calculate the approximations of the subtrajectory ending at $x_i$:
       $T_1 = (x^1_1, x^1_2, \ldots, x^1_{n_1})$ ,
       $T_2 = (x^2_1, x^2_2, \ldots, x^2_{n_2})$ ,
       ...
       $T_r = (x^r_1, x^r_2, \ldots, x^r n_r)$,
   where $r$ is the resolution level, $n_1$, $n_2$, … $n_r$ are not greater than the algorithm parameter $N$ and $x^1_{n_1} = x^2_{n_2} = x^r_{n_r} = x_i$.
5: Compute $Z(T_1)$, $Z(T_2)$, …, $Z(T_r)$ (zigzagging for empty or one-point degenerate trajectories is assumed to be 0).
6: If any of them exceeds $Z^{\max}$ Then
7:     Trigger alarm
8:     therewasanalarm = true
9: End if
```

The proposed algorithm is also depicted in Figs. 2–3.

An alarm is generated only once for a given route in order to avoid the operator's distraction by a storm of alarms for consecutive trajectory points. This feature uses the Boolean *therewasanalarm* variable.

Note that since variables n_1, n_2, …, n_r approximating sub-trajectories at particular resolution levels are limited by N, the number of resolution levels r is finite. Therefore, the algorithm conforms to the stream regime. The price that needs to be paid for this is that sub-trajectories may be approximated at N points at the most. In practice, it is difficult to implementing step 1 without storing all points x_0, x_2, …, x_i in the memory.

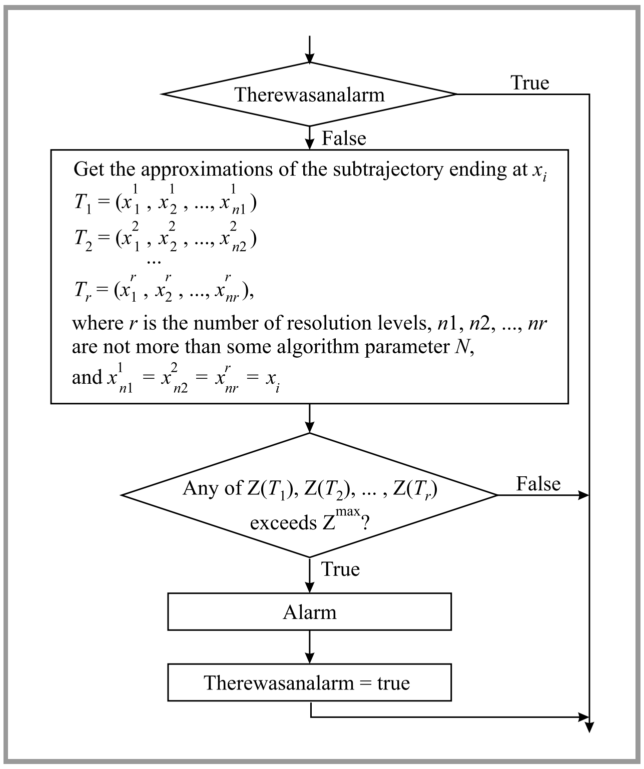
Therewasanalarm
True
False
Get the approximations of the subtrajectory ending at x_i
$T_1 = (x^1_1, x^1_2, ..., x^1_{n1})$
$T_2 = (x^2_1, x^2_2, ..., x^2_{n2})$
...
$T_r = (x^r_1, x^r_2, ..., x^r_{nr})$,
where r is the number of resolution levels, $n1$, $n2$, ..., nr are not more than some algorithm parameter N,
and $x^1_{n1} = x^2_{n2} = x^r_{nr} = x_i$
Any of Z(T_1), Z(T_2), ... , Z(T_r) exceeds $Z^{\max}$?
False
True
Alarm
Therewasanalarm = true

Fig. 2. Algorithm 1 used for processing a new trajectory point.

2.1. Selecting Route Approximations at Specified Resolution Levels

At each resolution level, the zigzagging of approximations $T_1, T_2, \ldots, T_r$ of the sub-trajectory is examined. These approximations are some trajectories made up of selected points of the sub-trajectory that has been observed so far. The selection of points and the construction of T_i is based on keeping memory usage limited.

Determination of T_i is based on characterization of the resolution, performed by the reference length step for a given i. This reference length is called $stride_i$. This is given by:

$$stride_i = minstride^{2i} \quad \text{for } i = 1, \ldots, r .$$

The algorithm's *minstride* parameter defines the length of the reference segment of the trajectory at the finest resolution level. Parameter r itself is evaluated as:

$$r = \left\lceil \log_2 \frac{maxroutelength}{rrbuflen \cdot minstride} \right\rceil ,$$

where $rrbuflen = N-1$ is the assumed maximum number of segments in T_i, $maxroutelength$ is the maximum possible length (sum of segment lengths) of the entire trajectory. The default parameter settings are: $rrbuflen = 4$, $maxroutelength = 20,000$ km, $minstride = 2$ km.

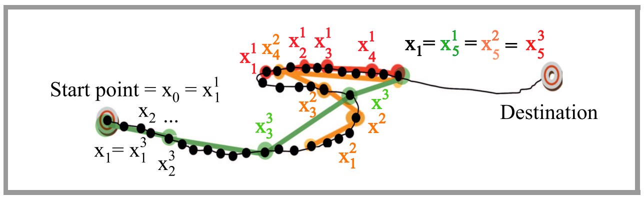

Fig. 3. Approximating the trajectory created so far with trajectories at 3 different resolution levels ($r = 3$). Probably, only the medium resolution trajectory t_2 (yellow) has a zigzagging measure that exceeds the alarm threshold.

Trajectories T_j for $j = 1,\dots,r$ are updated upon receiving a current point x_i of the trajectory according to Algorithm 2. Here, T'_j is a cyclic buffer storing a sequence of `rrbuflen` latest elements appended to this buffer. Ti'_j is initialized to empty sequence. Variable y_j is initialized to x_0.

Algorithm 2. Update of trajectory approximation T_j with a new trajectory point x_i

```
1: $stride_i := \mathrm{pow}(2i, minstride)$
2: $r := \lceil \log_2 maxroutelength/(rrbuflen \cdot minstride) \rceil$
3: **If** $\mathrm{dist}(x_i, y_i) \geq$ `stride`$_i$ **Then**
4:     Append $x_i$ to $T_j$ (with deleting all but rrbuflen last
       element of the sequence stored in $T'_j$.
5:     Set $y_i = x_i$.
6: **End if**
7: **Return** current $T_j$ as sequence stored in $T'_j$ (if it ends
   with $x_i$) or with appended point $x_i$ (otherwise).
```

In other words, sequence T_j is designed by choosing points of sequence x_i. The first is x_0, any new point x_i from the vehicle's trajectory is appended if and as soon as it is distanced from the previous appended point by at least $stride_i$. Such a sequence is appended the current point, if it does not end with it. At most last $rrbuflen$ appended points are finally taken to form sequence T_j. Such an approach ensures limited memory usage during the construction process, which is $\mathcal{O}(rrbuflen)$. Moreover, sequence T_j has segments of length of approximately $stridei_i$, except for the last segment which was added to keep sequence T_j as fresh as possible.
The settings for r and $stride_i$ (for $i = 1,\dots,r$) are selected so that:

- the logarithms of numbers `stride`$_i$ form an arithmetic sequence,
- the coarsest resolution level, T_r is capable of covering the entire trajectory of the maximum length $maxroutelength$, i.e. to contain its first and last point. The T_i for other (i.e. those used to assess local zigzagging of finer resolution levels), might obviously be shorter.

2.2. Extensions

The algorithm is equipped with several extensions. Different zigzagging limits may be used for $Z(T_1)$, $Z(T_2),\dots,Z(T_r)$, instead of a single $Z^{\max}$, in order to allow greater zigzagging at fine resolution levels. For given T_i, the limit is:

- $Z^{\max,\text{short}}$ if $stride_i < zigstridethreshold$,
- $Z^{\max,\text{long}}$ otherwise.

The reason for this extension is an observation that natural zigzagging of vehicles differs depends on the scale. In the case of routes that are less than 10 kilometers long, roads are often projected as being perpendicular to each other. Furthermore, drivers must perform maneuvers that increase the zigzagging measure, like bypassing one-way roads, making loops at multi-level intersections, obeying "no-turn" signs, etc. Hence, the default parameter settings are $Z^{\max,\text{short}} = 2.0$, $Z^{\max,\text{long}} = 1.7$, $zigdistthershold = 10$ km.
The second extension allows to include the destination point in the trajectory that has been covered so far. If we know the trip's destination (which is usually the case), we may not only examine various zigzagging measures based on the sub-trajectory covered so far, but we may also proactively check whether various zigzags will need to become excessive in the future, assuming that the vehicle will eventually reach its destination. This extension works as follows. Upon receiving of consecutive point x_i, Algorithms 1 and 2 are invoked, but:

- the zigzagging excess check is performed as in Algorithm 1 (possibly with the extension of various zigzagging limits),
- the zigzagging check is performed again, not for approximate trajectories $T_1,\dots,T_r$, but for these trajectories, each with the destination point appended.

If any of these checks triggers an alarm, the entire algorithm generates an alert. This extension allows to trigger alarms in some situations. A typical situation is when a vehicle starts its trip in the direction that is opposite to its destination, in order to perform some illegal activity. Then, in the basic algorithm, any zigzagging activity observed increases extensively only after the illegal location has been reached and upon the vehicle starts moving towards the legal destination. However, an operator observing the route might be aware of the bad intentions of the driver even before the illegal location is reached. The vehicle driving in the opposite direction it allowed to continue doing so. By adding an artificial segment to the route covered so far, which predict the route ahead, allows the proposed algorithm to alert the operator before the illegal location is reached. This advantage is illustrated by one of the experiments presented further on.

3. Discussion

Parameter *minstride* of the finest resolution naturally determines the rank of distance corresponding to "nonsensitivity level" of the algorithm to small deviations from the route. Along short sections of the trajectory, comparable with minstride, zigzagging may reach high values without triggering an alarm. This allows the driver to safely deviate to roadside facilities, such as hotels, restaurants or service stations. This mechanism allows for some other maneuvers as well: passing multi-level intersections, bypassing local one-way roads, performing maneuvers required by road signs. These trip disturbances may cause considerable local trajectory zigzagging, even imply trajectory looping, but as long as they are limited to a specific area of the *minstride* rank, they will be accepted by the algorithm.

Memory usage. Memory usage of the presented algorithm is $\mathcal{O}(rrbuflen \cdot r)$, where

$$r = \left\lceil \log_2 \left(\frac{maxroutelength}{rrbuflen \cdot minstrade} \right) \right\rceil . \tag{2}$$

The proposed algorithm assumes a limited length of the route and *rrbuflen* and *minstride* are arbitrarily chosen constants. While limiting the route length by constant *m*axroutelength, constant memory usage is ensured. This assumption is not very restrictive in practice and memory usage increases only logarithmically with a potential increase in *maxroutelength*.

Metrics used. In the algorithm, the length of shortest line on the Earth surface between two points x, y is computed by function $\mathrm{dist}(x,y)$. However, there is a possibility of using map information instead, and of calculating the values of $\mathrm{dist}(x,y)$ as the shortest road-based route between x and y. Some additional care must be taken to account for potential noncommutativity of such a distance. This increases the precision of the algorithm and boosts its independence from specific road network configurations. Certainly, the zigzagging limits would be decreased in such a scenario. However, such a solution is complex to implement in large-scale systems, i.e. for monitoring purposes. It requires additional memory to cache the map data locally in order to avoid extensive data traffic exchanges with the map server. Moreover, cache memory should also offer a mechanism for quickly calculating the distance between specific locations. Finding the shortest route online would clearly consume too much resources. The mechanism should perhaps contain some mesh of points with predefined distances and, based on this data, should approximate the distance between two given locations. This an interesting research problem in itself.

Traveling salesman case. It is often the case that a vehicle starting its trip has several destinations to visit. The algorithm may still be applied to such a scenario, even when the order of the visits is not known to the algorithm. The trajectory covered so far needs to be simply reset to empty and started anew upon any destination. This will allow only the zigzagging along the route section between two consecutive destination points to be examined. This fragment should be economical, i.e. no excessive zigzagging should be present. The extension expanding trajectories by destination locations is slightly more difficult to adopt in this case, since there are many destinations and we do not know which is to be reached first. Therefore, at the receiving point x_i, we should try all destinations and select the scenario that is most favorable for the driver. If no attempts trigger the alarm, the algorithm should keep quiet.

4. Numerical Demonstration

Several experiments with the proposed algorithm have been conducted. Their aim was to simply validate the presented approach, and to showcase the specific features and the behavior of the algorithm in typical scenarios. The extension providing various zigzagging limits was on, while the extension making use of the destination location was off by default, and was only switched on in one experiment. Default parameter settings were used.

A Java environment was used in the experiments. The code was written to retain the algorithm's efficiency, i.e. cyclic buffers were used to collect the selected past points of the trajectory.

The routes were generated in using the Google MyMaps service as optimal routes form the town of Skierniewice to Warsaw (Poland). To obtain uneconomical variants of the route, specific mid-points were added to Google's route finder. These points served as illegal locations, while the algorithm was still simply aware of Warsaw as the vehicle's destination. The algorithm observed the trajectory online and triggered alarms, detecting suspected fraud. It needs to be remained that the algorithm is designed to alert once for each route to avoid a storm of alarms for consecutive trajectory points. The point of the trajectory at which the alarm occurred was registered and included in the subsequent presentation of the results.

The following experiments have been performed:

- the route from Skierniewice to Warsaw without any mid-points. It was expected that this route would not be assessed as suspicious;
- the route from Skierniewice to Warsaw with an additional mid-point in a forest in the village of Krakowiany, near the main road. This route includes an apparent deviation using a road leading to the forest and should trigger an alarm. The forest, however, is located near the main road (relatively to the length of the entire route). Thus, zigzagging measure along

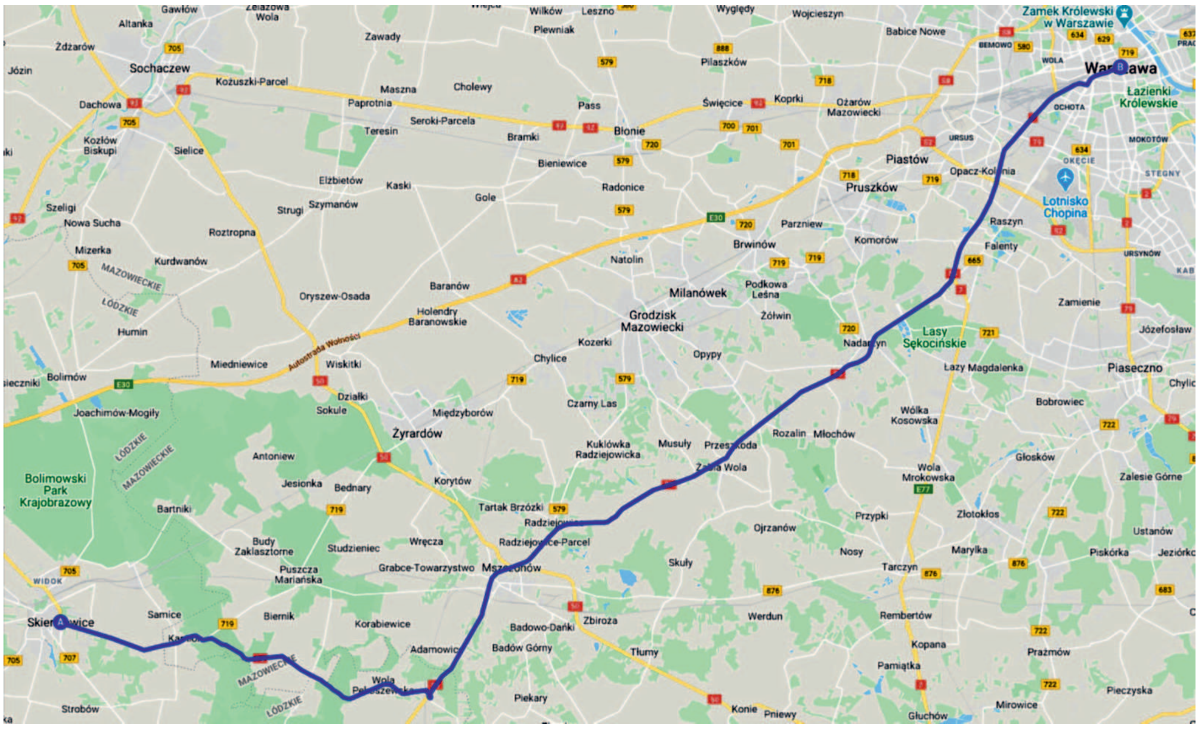

Fig. 4. Route from Skierniewice to Warsaw without a deviation – no alarm triggered.

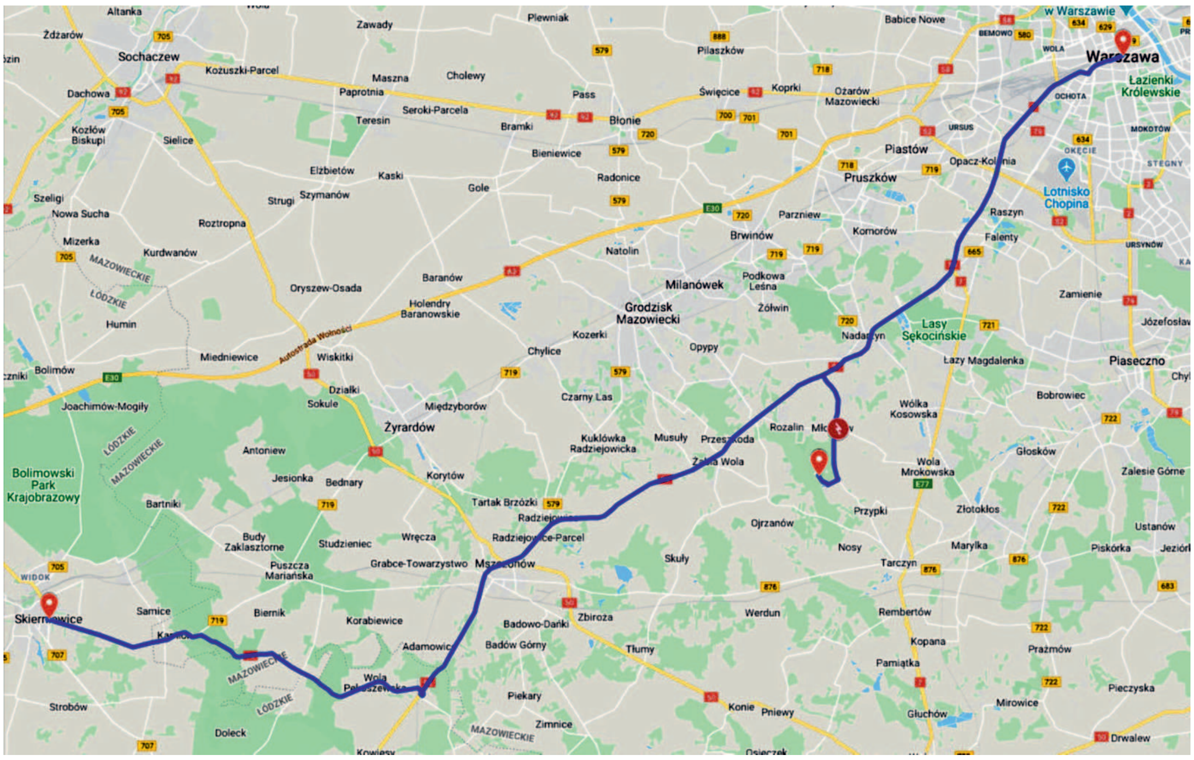

Fig. 5. Route from Skierniewice to Warsaw with a deviation to a forest in Krakowiany. An alarm has been triggered on the local road leading to Krakowiany, after visiting the village (during the return from the forest).

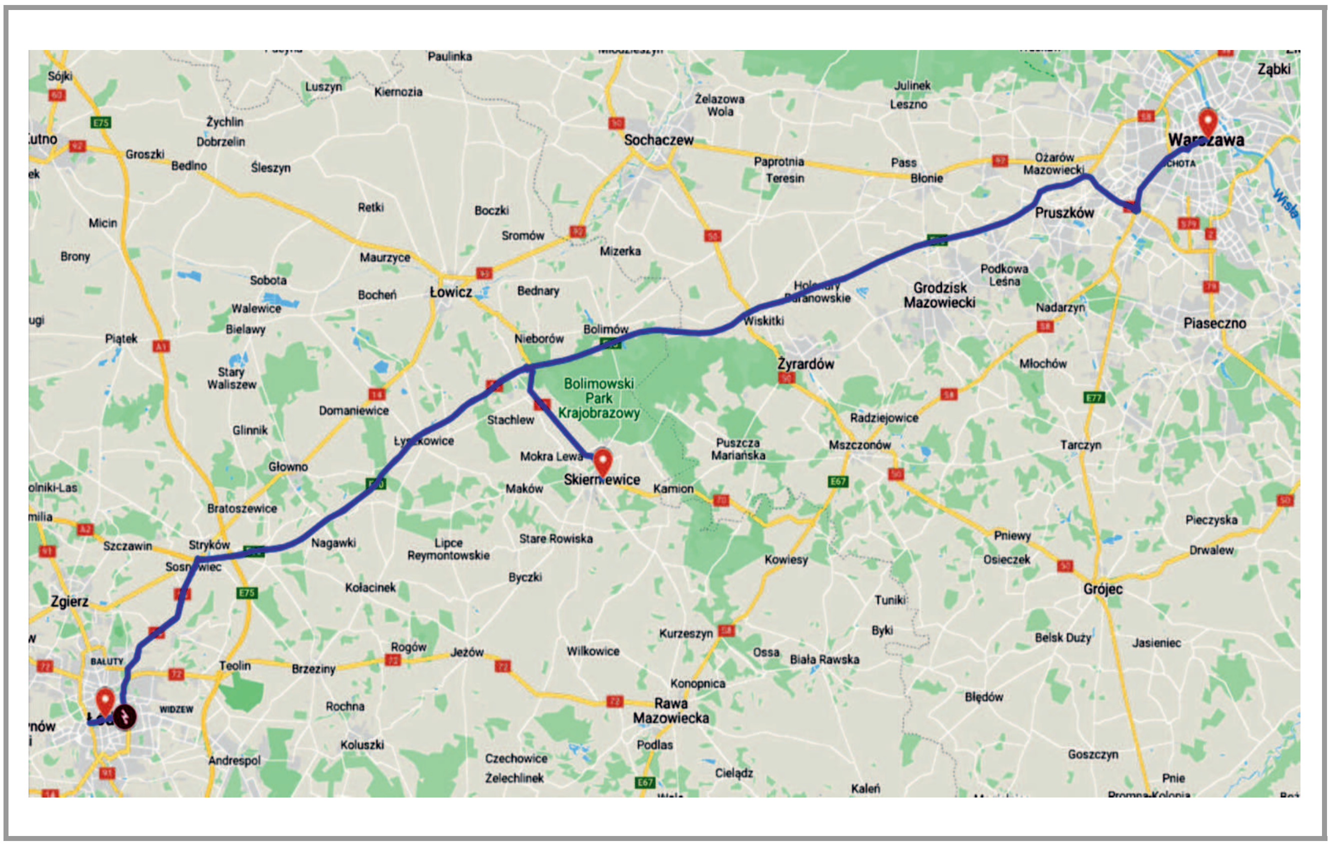

Fig. 6. Route from Skierniewice to Warsaw with a deviation to Łódź. The alarm has been triggered after visiting Łódź, on the return leg of the journey.

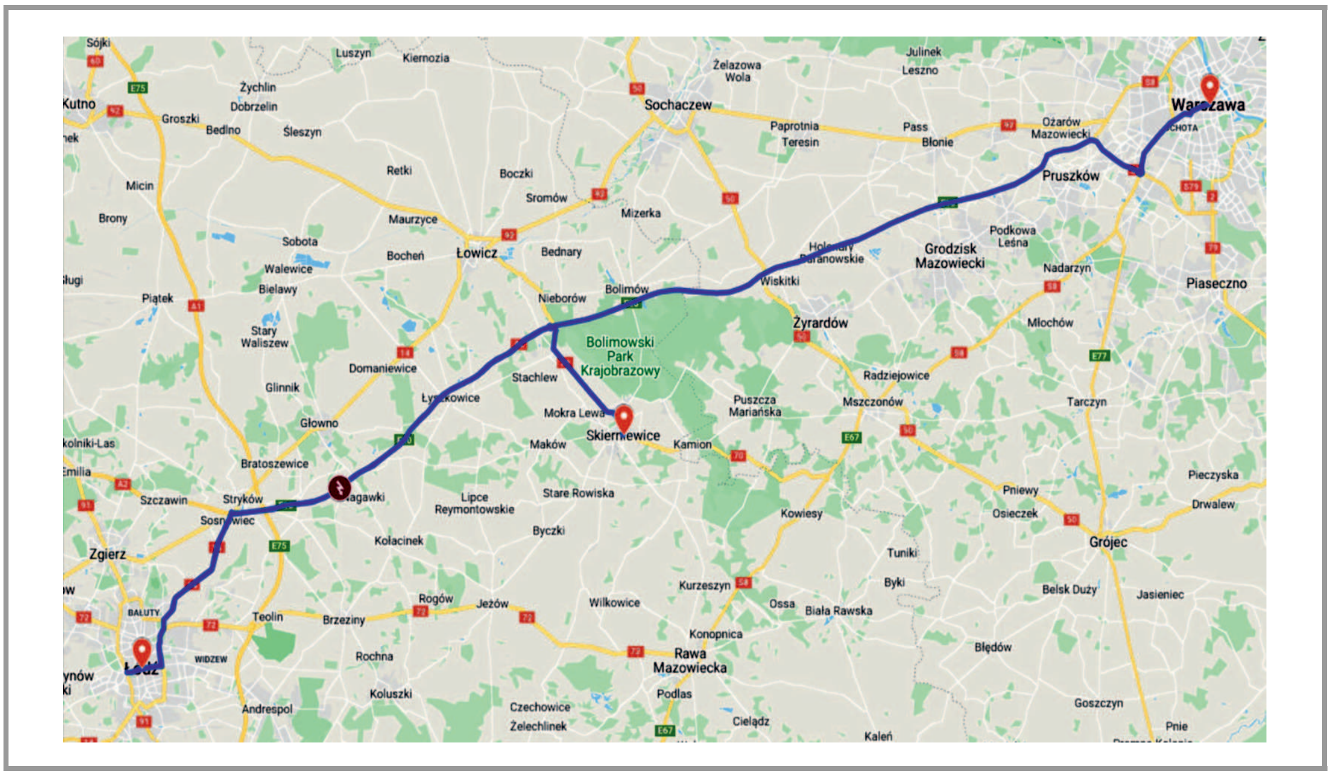

Fig. 7. Route from Skierniewice to Warsaw with a deviation to Łódź. The algorithm uses route augmentation, making use of the knowledge of the destination. The alarm was triggered before reaching Łódź.

the entire route could not be excessive. The example checks whether the algorithm reacts to an increase in zigzagging observed locally;

- the route from Skierniewice to Warsaw, with a mid-point of Łódź. When going to Łódź, a driver leaving Skierniewice needs to go in the opposite direction than when heading to Warsaw. Therefore, such a route should trigger an alarm;
- the route from Skierniewice to Warsaw, with a mid-point of Łódź and, unlike in the case of other experiments, the knowledge of the destination has been used in this test. This means that the algorithm augmented the trajectory covered so far with a segment between the current location and the destination. This augmentation should result in the alarm being triggered even earlier than in a scenario without augmentation.

The routes are presented in Figs. 4–7. The results of the experiments are presented in the form of icons in pink circles positioned at the locations where an alarm was detected. Since the roads on which the alerts were triggered are of the two-way variety, it is difficult to see, in the pictures, which direction the vehicle was driving in while causing the alarm. This is stated in the captions.

The alarms were triggered in concordance with our expectations. They appeared precisely along the routes with mid-points that emulated illegal load or drop locations. The route with the deviation to a forest shows the algorithm's ability to consider local route zigzagging. As far as the deviation to Łódź is concerned, the advantage of augmenting the route with a segment between the current location and the destination was shown. Without the augmentation, the alarm is triggered after visiting the town. With the augmentation, the alarm is generated earlier, on the way towards Łódź. This is beneficial, since the algorithm with augmentation is capable of making the same observations as a human being. The vehicle is supposed to head for Warsaw, but drives in the opposite direction instead. Also, the alarm is raised, in such a scenario, before the illegal activity takes place, which allows the proactive actions to be taken by the operator.

5. Conclusions

The proposed stream algorithm is capable of detecting truck fraud. In contrast to its numerous counterparts, the proposed solution uses an adequate and human-understood route economy measure. It is lightweight and does not require a learning phase. In experimental trials it turned out to behave reasonably and predictably.

References

[1] S. S. Dukare, K. Rane, and D. A. Patil, "Vehicle tracking, monitoring and alerting system: a review", *Int. J. of Computer Applications*, vol. 119, no. 10, pp. 39–43, 2015 (DOI: 10.5120/21107-3835).

[2] "PUESC" [Online]. Available: http://puesc.gov.pl

[3] M. Saravanan, S. Aishwarya, and L. N. Aravindan, "Tracking anomalies in vehicle movements using mobile GIS", in *2013 Science and Information Conf.*, London, 2013, pp. 845–852 [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6661840

[4] A. Kuepper, U. Bareth, and B. Freese, "Geofencing and background tracking – the next features in LBSs" in *41. Jahrestagung der Gesellschaft für Informatik, INFORMATIK 2011 – Informatik schafft Communities, Berlin, Germany, October 4-7*, 2011 [Online]. Available: https://www.user.tu-berlin.de/komm/CD/paper/010221.pdf

[5] F. Reclus and K. Drouard, "Geofencing for fleet & freight management", in *2009 9th Int. Conf. on Intell. Transp. Syst. Telecommun., (ITST)*, 2009, pp. 353-356 (DOI: 10.1109/ITST.2009.5399328).

[6] P. Deshmukh, A. Bhajibhakre, S. Gambhire, A. Channe, and N. Deshpande, "Survey of geofencing algorithms", *Int. J. of Comp. Science Engin. Techniques*, vol. 3, no. 2, 2018 (DOI: 10.29126/2455135x/IJCSE-V3I2P1).

[7] R. R. Sillito and R. Fisher, "Semi-supervised learning for anomalous trajectory detection", in *Proc. of the British Machine Vision Conf.*, 2008, pp. 103.1–103.10 (DOI: 10.5244/C.22.103).

[8] Y. Ge, H. Xiong, C. Liu, and Z. Zhou, "A Taxi Driving Fraud Detection System", in *11th IEEE Int. Conf. on Data Min.*, Vancouver, Canada, 2011, pp. 181–190 (DOI: 10.1109/ICDM.2011.18).

[9] J. B. Oliv, "Anomaly detection and modeling of trajectories", M.Sc. thesis, CMU-CS-12-133, School of Comput. Sc., Carnegie Mellon University, Pittsburgh, 2012 [Online]. Available: http://reports-archive.adm.cs.cmu.edu/anon/2012/CMU-CS-12-133.pdf

[10] Y. Bu, L. Chen, A. W. Fu, and D. Liu. "Efficient anomaly monitoring over moving object trajectory streams", in *Proc. 15th ACM SIGKDD Int. Conf. on Knowledge Discov. and Data Mining*, Paris, France, 2009 (DOI: 10.1145/1557019.1557043).

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, 2nd ed.* London: MIT Press, 2001 (ISBN: 9780262032933).

Paweł M. Białoń is with the National Institute of Telecommunications. He received his Ph.D. in Automatic Control and Robotics from Warsaw University of Technology in 2013. His scientific interests include decision support and optimization with applications in telecommunications and data mining.

https://orcid.org/0000-0002-7781-9212
E-mail: P.Bialon@il-pib.pl
Advanced Information Technologies Department
National Institute of Telecommunications
Szachowa 1
04-894 Warsaw, Poland