UDC 004.021

D.Sci. Tech. **Zaslavskyi V.A.** (TSNUK)
Pushkarenko Yu.V. (TSNUK)

DOI: https://doi.org/10.17721/2519-481X/2021/73-03

PRINCIPAL CURVE TRAJECTORY ANALYSIS

Increasing availability of probe data sources gives an opportunity to use the data in automatic map creation process, refine the shape of existing maps as well as potential for generating source of truth (i.e., ground truth data) in military as well as for civil aspects. That is needed for confirming the quality of existing maps as well as compilation them from different sources to achieve the comprehensive result of creating very high-definition maps. The main reason is to compile the resulting map not only from satellite imageries but also using another source of compilation or confirmation. We don't do deep dive in map compilation process itself but concentrating on map-matching problematic from perspective of GPS Probes trajectories which is very noisy by nature. The present paper proposes an analyzing of methods based on principal curve trajectory collected from raw GPS positions. Probes positions itself from phone inside a car are noisy, they don't necessarily match the actual position of the car for a given moment. Assuming the car drives on a street according to regulations, the raw position can be matched to street locations by means of a map matching algorithm. Using a series greatly improves the stability and plausibility of the map matched positions, especially when probes samples are noisy or sparse and different roads are close together (e.g. crossings, bridges, tunnels, slip roads) it could be useful for creating bi-directional road geometry from sparse probes. The resulting road segments in the road network graph enable conflation with existing map data to identify map changes including base maps.

Trajectory analysis and related algorithms have recently attracted substantial attention, thanks to technological advances in navigation and mapping systems. Nevertheless, some fundamental concepts are still lacking a thorough study. The identification of a middle (representative) trajectory in a bundle of trajectories is one of them. Without conscious reasoning, a middle trajectory is a trajectory that lies in the middle of a collection of trajectories. However, this definition is far from being comprehensive.

This research work is focused on the concept of finding a principal curve trajectory among a bundle of trajectories with specific source and destination points. The main idea is to use the timing information associated with trajectories to improve existing methods for trajectory analysis. We investigate the concept of a principal curve related to timing information, we give a review of algorithms for all existing methods, analyze the worst-case running time, and show that under certain assumptions methods such as timing can be implemented efficiently.

Key words: Map-Matching, Geo-Spatial Analysis, Computational Geometry, Cluster Analysis, Probe Data, Road Geometry Extraction, Road Maps, Spatial Data Mining.

Introduction. A formal definition of trajectory is specified in [1] as a time-stamped path taken by a moving entity, represented by a sequence of n tuples of points and time stamps (p_0, t_0), $(p_1, t_1), \dots, (p_{n-1}, t_{n-1})$. Points have spatial and temporal components. The spatial component typically represents a two or three dimensional space. Here, we assume that the space is two dimensional. A bundle of m distinct trajectories $T_0, \dots, T_{m-1}$ with the same start and ending points, therefore results in an input size of $\theta(nm)$.

In an ideal situation, the time stamps of all trajectories in the bundle are exactly the same, but this is usually not the case. Generally, trajectories are collected with different or irregular sampling rates, at different times, and data can be missing as well. In between time stamps, we have no information about the actual movement path of the entity. The standard assumption can be that the entity moves with constant velocity from a time-stamped point to the next time-stamped point in a straight line. This assumption leads to an approximation of the actual data, which becomes more inaccurate when sampled at longer intervals. As a result, the path of a trajectory is considered as a polygonal curve with n edges that can self-intersect, and can have repeated vertices at the same location if the entity stands still. The number of points defining a trajectory is usually much larger than the number of trajectories in a bundle $n \gg m$.

Analysis of previous studies. Various methods for trajectory analysis have been developed in different fields of science including computational geometry and data mining. Trajectory data sets can be analyzed in a variety of ways. They are usually clustered into a collection of subsets that have a high similarity regarding (a) certain property(-ies), such as location. Nevertheless, processing large amounts of trajectory data is a challenge. Trajectory data compression can be regarded as a solution to address this problem, particularly for the trajectories with huge sampled data points to improve the efficiency of computations. This compression can be carried out for individual trajectories, but in most applications, the data contains similar bundle of trajectories in terms of space and/or time, and as such, an alternative method would be to compress a trajectory bundle to represent similar trajectories with a single representative trajectory. We call this representative trajectory a middle trajectory without loss of generality. Furthermore, using a single representative trajectory enables processing data in a simple, robust and more predictive way. As an example, a representative trajectory for an specific route can be used to predict where a vehicle will be at a certain time. Alternative applications of middle trajectories include clustering and visualizations. As an example, a middle trajectory can act as the medoid in $k - medoid$ clustering. In the context of visualization, instead of showing a large collection of trajectories, one may identify subsets of similar ones, and replace them by a middle trajectory whose width is determined by the size of the subset.

Formation of the problem. Given a set of m trajectories $T = \{T_0, \dots, T_{m-1}\}$ with the same source and destination points s and d, we want to find a representative trajectory T_r that minimizes a defined distance function to all trajectories $T_i, 0 \leqslant \mathrm{i} < \mathrm{m}$ in the set and T_r contains points from input trajectories. In case not specified otherwise, the distance function is defined as the Euclidean distance between trajectories.

A representative trajectory has to be in the middle of all trajectories in the bundle. However, we first have to define what is considered middle. There are two main types of interpretations of what a middle trajectory is for a set of input trajectories: *median* and *mean* trajectory [2]. They can produce a trajectory that is in the middle, regarding space or time, or both. There could be different interpretations for the definitions of mean and median here compared to the standard arithmetic definitions of these parameters, however the general intuition remains the same. The main difference between a mean and a median trajectory is that the latter uses only points of the input trajectories. An alternative definition for a median trajectory is to use only the edges of the trajectories present in the input set, or parts of the edges, and switch at intersections [1]. Both mean and median trajectories have their own applications. Mean trajectory which is based on the spatial component of the input trajectories looks more accurate compared to a trajectory that is restricted to one of the inputs. Nevertheless, there can be a strong reason why one prefers a median trajectory over the mean one. A typical example would be for moving objects that have to avoid obstacles. The same property should also hold for the middle trajectory. If no information about the obstacles is known in advance, then the only way to ensure that the representative trajectory crosses no obstacle is to use parts of the original input trajectories. On the other hand, if such information is known, then it may also be possible to use this information and make the mean path go around the obstacles. Fig. 1 shows examples of mean and median trajectories. The median one follows the existing trajectories and thus avoids potential obstacles if any is defined. The mean trajectory does not necessarily follow the input trajectories and may pass through obstacles. One major disadvantage for mean trajectory is its sensitivity to outliers. Data sets of input trajectories containing substantially diverging measurements will influence a calculated mean trajectory. Fig. 2 shows an example of this phenomenon.

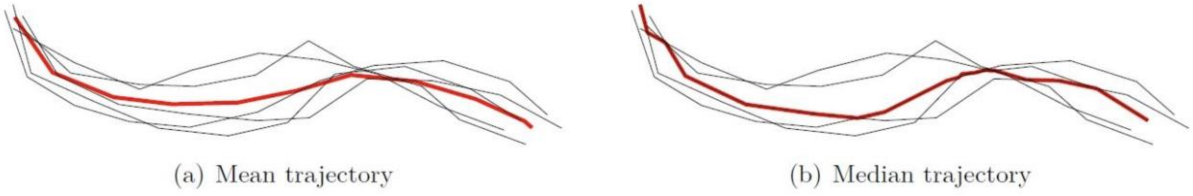

Figure 1 - Examples of (a) Mean trajectory (b) Median trajectory

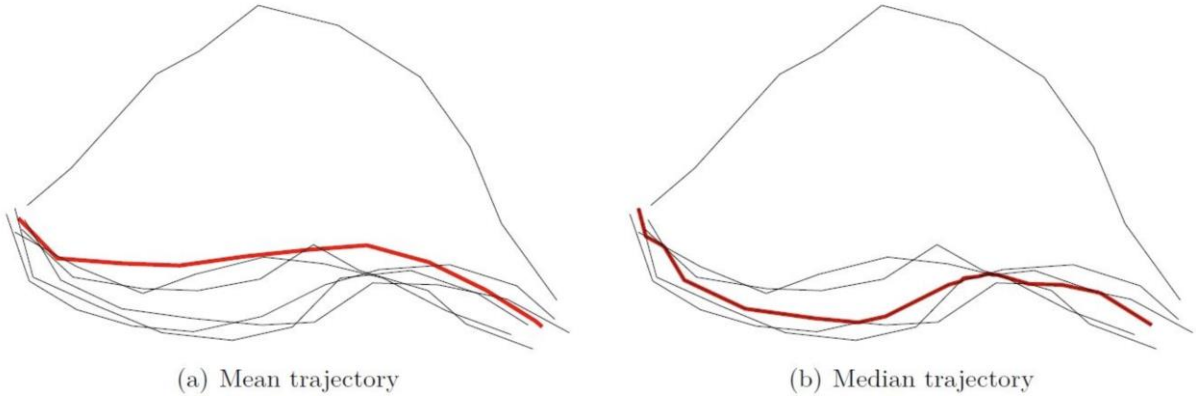

Figure 2 - Effects of outliers on (a) Mean trajectory (b) Median trajectory

Trajectory data usually include a temporal component in addition to an spatial component, however in most cases the temporal component plays no role in computation of the middle trajectory. The reason is that it would be difficult to utilize this information in an efficient way due to inconsistencies between data points that is mostly present in a trajectory bundle. As an example, suppose the trajectories are from different vehicles on a specific route. The data could be collected on different days and times, as a result, we cannot directly use the temporal component. Even if vehicles had exactly the same starting and ending locations and take more or less the same path, we cannot simply align the starting times of the travels, because one vehicle may have been held up due to traffic conditions which distorts the time correspondence that we set up at the starting point. Thus, in many situations, a middle trajectory is solely processed based on the spatial component and considers the path (shape) of the trajectories.

Main Part. So far several properties have been proposed for a representative trajectory in literature. Nevertheless, most of these properties are not well defined, but rather intuitively assumed to hold for a representative trajectory. Here is a list of these properties:

- continuous: a contiguous curve from a source to a destination without interruptions
- central: locally central within the bundle regarding the positions of the input trajectories
- comparable length: its length should be more or less the same as the input trajectories
- comparable angular change: its total angular change should be more or less the same as input trajectories
- comparable vertices: its total number of vertices should be more or less the same as input trajectories
- maintain edge direction: for every edge of input trajectories included, the direction should be retained accordingly
- follow the majority: should visit regions only if most of the input trajectories do
- consist of input trajectories: should be composed of points of input trajectories

It addition it is desirable that a representative trajectory be robust against outliers and special cases and there will be efficient algorithms for its computation.

Several approaches have been used to identify a representative trajectory for a set of clustered trajectories. Buchin et al. [1] present two different methods to construct a median trajectory for a set of input trajectories. The first method is based on simple definition of mean level in an arrangement of lines while the second method uses the concept of homotopy with respect to sufficiently large faces in the arrangement formed by input trajectories. Both methods produce a trajectory that consists of pieces of the input and ignore the temporal component of trajectories. The simple median trajectory starts on the middle trajectory and always switches to another trajectory at an intersection point. If all the m input trajectories are ordered based on the starting point s (in the outer face of the arrangement of curves) with the first and last trajectories adjacent to the outer face, then the middle trajectory starts

from the $\left[\frac{m}{2}\right]$ -nd edge in the order. It is shown that under certain conditions, such a median is always in the middle. They prove that the median trajectory satisfies the property that for any point p must cross to reach the unbounded face (including the one(s) on which p lies) is $\left[\frac{m+1}{2}\right]$. In presence of outlier trajectories, they shall be excluded from the total number of trajectories. The simple median trajectory has a upper bound time complexity of $O((nm)^2)$ because of the total $n.m$ line segments that should be processed, where m is the number of input trajectories and n is the maximum number of data points for any trajectory. For practical situations, the time complexity is improved to $O((nm + k)a(nm)\log(nm))$, where a is the inverse Ackerman function and k is a number of edges in the output trajectory.

The main problem with this approach is that it will most probably fail to find a representative trajectory when the trajectories are self-intersecting. An example is shown in Fig. 3. All the three input trajectories go through a loop but the median one does not follow a similar path. The problem happens because at some intersection, an edge is selected which is not a part of the direct continuation of the path, rather part of a trajectory on the way proceeding the loop. A similar problem can even emerge when the trajectories are not self-intersecting. As shown in Fig. 4, the usual path of the majority of the input trajectories is not followed accordingly. In this example, two of the input trajectories make a kinda detour, while a third one takes a straight path towards the destination. The median trajectory follows the unwanted straight one.

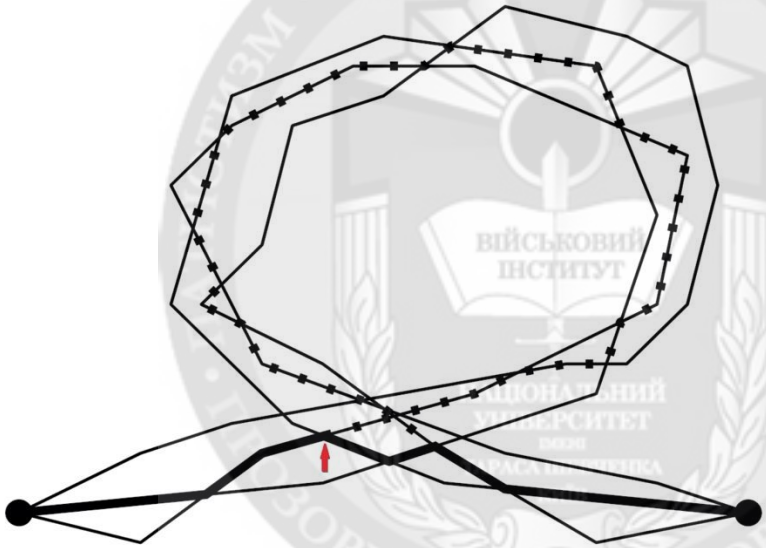

Figure 3 - Self-intersecting trajectories can perform undesirable for the computation of the median trajectory

Unlike the simple median trajectory, the homotopic median does not necessarily switch at every intersection points. When trajectories self-intersect, a point p is placed in the face(s) around which it loops. Switching trajectories is only done if the median calculated up to the switching point is of the right homotopy type, determined by the signature of the (sub)trajectory. This signature is the intersection of the trajectory with the vertical line through each point p and side of the intersection (i.e. above or below

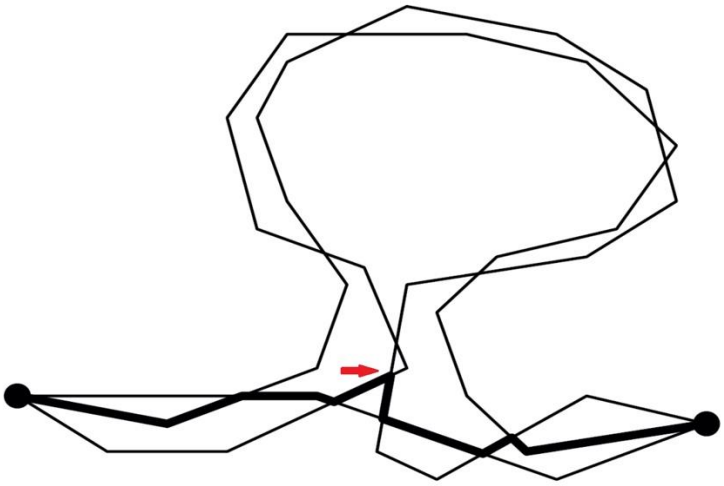

Figure 4 - The median trajectory can miss part of the path of the input trajectories with no self-intersections

Unlike the simple median trajectory, the homotopic median does not necessarily switch at every intersection points. When trajectories self-intersect, a point p s placed in the face(s) around which it loops. Switching trajectories is only done if the median calculated up to the switching point is of the right homotopy type, determined by the signature of the (sub)trajectory. This signature is the intersection of the trajectory with the vertical line through each point p and side of the intersection (i.e. above or below p with the values p^+ and p^-, respectively). Fig. 5 shows two examples of input trajectories and their corresponding homotopic types.

Figure 5 - (a) three input trajectories that loop around a face of the subdivision in which p is placed. All three trajectories have the signature $p^- p^+ p^-$. (b) two trajectories that are homotopic, with signature $p_1^- \; p_2^- \; p_3^+ \; p_3^- \; p_2^+ \; p_2^- \; p_3^-$

An example of seven input trajectories is shown in Fig. 6 (a). Most of the trajectories follow the path numbered from 1 to 14. Five trajectories skip point 7 and six skip point 11. One trajectory skips points 5–8. One expects the representative trajectory to follow what most input trajectories do, and consecutively go from 1 towards 2, 3, 4, 5, 6 = 8, 9, 10 = 12, 13, and 14. The simple and homotopic median trajectories are marked in Fig. 6(b). As seen in the figure, the homotopic median (red) appears to give a more appropriate middle trajectory compared to the simple median one (green).

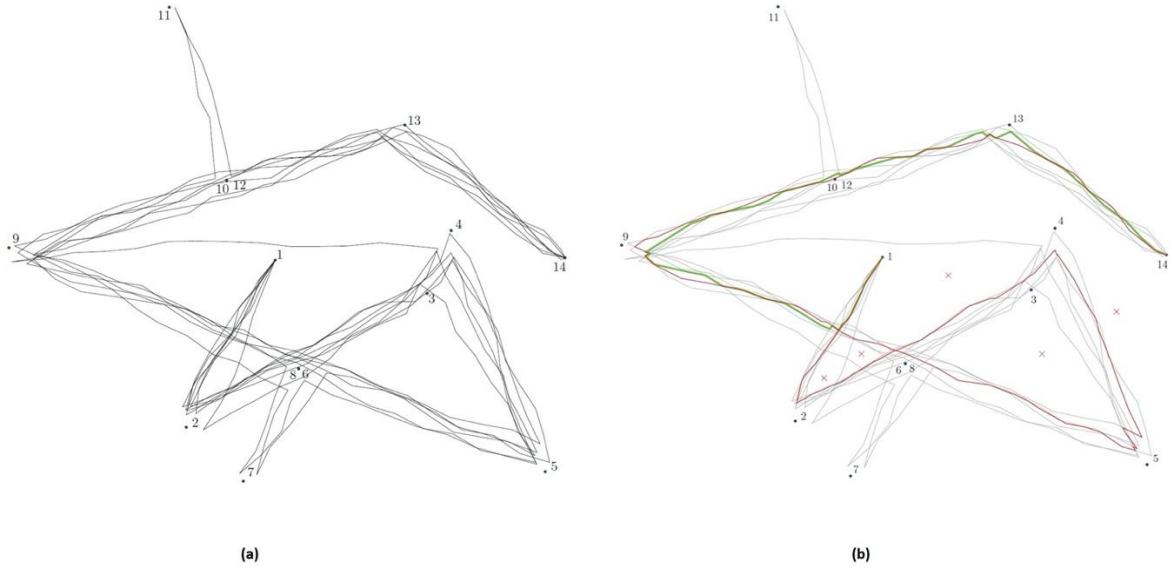

Figure 6 - **(a)** A set of input trajectories, generally following the numbered points excluding 7 and 11, **(b)** The simple median trajectory is marked with green and the homotopic median is marked with red

Although homotopic median appears to give intuitive median trajectories in many situations, there are two clear cases where it fails to identify a suitable representative trajectory. In [3], van Kreveld and Wiratma give examples where the homotopic median trajectory fails. In one case, if all trajectories make a detour that is back-and-forth over the same path, then no relatively large face is present to place a point in. Such a detour is incorrectly ignored by the homotopic approach. Fig. 7 illustrates this situation. Another case is when all trajectories are of a different homotopy type (e.g., when no large subset of homotopically equivalent trajectories exists), which results in one of the input trajectories being the median. Using a single input trajectory as the median might result in the computed trajectory to include regions that are visited only by a single input trajectory. This situation is illustrated in Fig. 8.

Figure 7 - The homotopic median approach will fail to make the detour because there is no large enough face to place a point

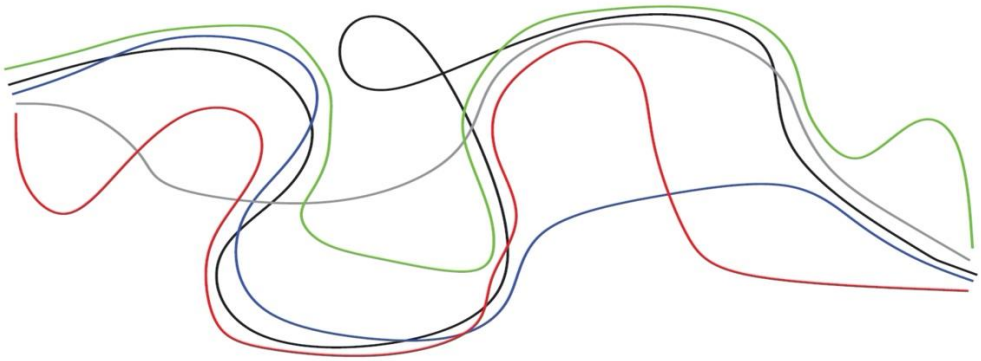

Figure – 8 Input trajectories with different homotopic types make the identification of a representative trajectory fail. There is no typical set of homotopically equivalent trajectories, thus one of the inputs would be identified as a median trajectory

The time complexity of computing homotopic median trajectory is: $O((nm)^{2+e}), e > 0$.

In [3], van Kreveld and Wiratma present a completely different method that computes median trajectories as an alternative to overcome the aforementioned drawbacks for simple and homotopic trajectories. This method is based on the idea that not all segments (edges) in T are suitable to be a part of the median trajectory T_r in case they are not close to the majority of trajectories in T. This implicitly implies that the edges which lie somehow in the middle of a bundle of trajectories are more eligible to be included in the median trajectory. As seen in Fig. 9, based on the majority definition, the dark green edges are most likely to be selected as part of the median trajectory, while red eges having the least chance.

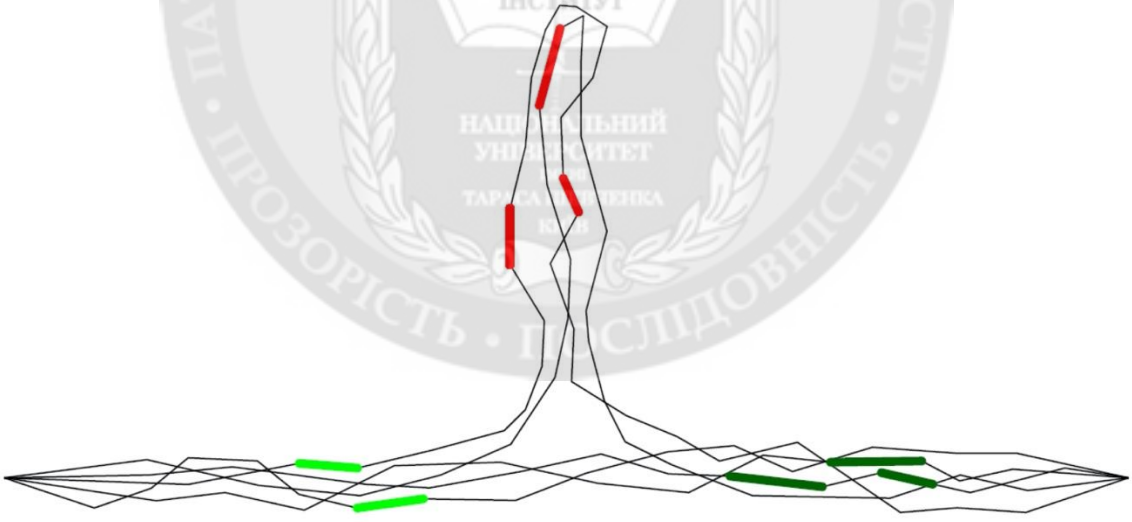

Figure 9 - Red edges are not good candidates to be part of a median trajectory; dark green ones are suitable due to their positions (close to the majority and somehow in the middle); being on the outside border of the bundle, the light green ones are less suitable compared to the dark green segments

In the proposed algorithm, aka *buffer median* algorithm [4], an edge e is called *useful* if a buffer of size δ around intersects with at least half of the total number of trajectories (the majority). Only if an edge e is considered useful then it is allowed to be part of the median trajectory, and subsequently an attempt is made to include e or some other edge in its neighborhood in the median trajectory. A rough high-level description of the majority median algorithm is that all the edges that are not useful

are deleted initially and for all the remaining useful edges an investigation is carried out to determine whether or not they can be part of the median trajectory. To this end, suppose that a directed planar graph φ is constructed with all vertices of the trajectories T plus all intersection points of the edges of the trajectories. The edges in φ are edges or sub-edges (intersections are also included) from the trajectories in T. The direction of the edges in φ are identical to the directions of edges in T. For simplicity, it is assumed that all trajectories of T start in a common vertex s, the source, and they all end in a common vertex d, the destination. As mentioned previously, this restriction can easily be removed by introducing two dummy vertices if needed. The median trajectory is assumed to be some directed path in φ from s to d only containing useful edges, but not all of the useful edges. To find such a path, all edges of φ that are determined not useful are removed initially. Then, the shortest path from s to d in φ is computed as the first (tentative) median. Given a median, a useful edge is called happy if it is in the median or within distance at most 2δ from an edge in the median. As long as there are unhappy useful edges, a better median may be constructed by including more unhappy useful edges. A useful edge that is "most unhappy" (the one that is furthest from the median) is selected (edge e), and let $S(e)$ be the set of edges of φ within distance δ from e. The shortest paths from s to all endpoints of edges in $S(e)$, and shortest paths from these endpoints to d, are computed and they are combined to find an overall shortest path from s to d that uses some edge of $S(e)$. If such a path exists, then this is the new median and e and all edges in $S(e)$ must be happy. If such a path does not exist, then it is not possible to make e happy when using only useful edges for the median. Hence, e and all other edges in $S(e)$ are ignored. The algorithm stops when all useful edges are happy or ignored. Whether or not a new median is computed, there may still be not ignored, unhappy edges elsewhere, so to proceed, the most unhappy edge for the latest median is located, and the same procedure is followed to try to include it in the median.

The median trajectory is supposed to pass through a sequence of sets of endpoints $P_1, P_2, \dots, P_k$ between s and d, where each P_i corresponds to the endpoints of some set $S(e)$ of edges in the algorithm. The order $P_1, P_2, \dots, P_k$ is determined incrementally during the algorithm using a voting strategy, which is robust against deviations, shortcuts and noise. Once the order of $P_1, P_2, \dots, P_k$ is settled, a shortest path is computed between s and d, using Dijkstra's algorithm in an iterative manner for all the intermediate endpoints. The final shortest path from s to d visits at least some node of each of $P_1, P_2, \dots, P_k$ in the specified order. The details of the algorithm and its pseudo-code is described in [4].

The time complexity of the algorithm is polynomial regarding the input size $m \times n$. Nevertheless, the worst-case running time is rather high. A worst-case analysis does not characterize the actual running time well because it would refer to hypothetical cases where all edges of a trajectory intersect all edges of all other trajectories. Under assumption that any edge of any trajectory intersects with constant number of edges of other trajectories (i.e. there are $O(nm)$ edge-edge intersections in total), the worst-case time complexity would be $O(n^3 m^3 log(nm))$. This assumption leads to a size of the graph that is linear in the input size, $O(nm)$, and it also implies that the computed median has at most $O(nm)$ vertices [3].

The majority median trajectory for the bundle of trajectories depicted in Fig. 6(a) is shown in Fig.10. The extra paths regarding points 7 and 11 are skipped by majority of trajectories. The homotopic median (depicted in Fig. 6(b)) and the majority median both have identical global shape that respects the majority of input trajectories. It is worth noting that the homotopic median is based on only four trajectories because three do not have the same homotopy type. Consequently, the homotopic median does not always lie in the middle. Furthermore, the homotopic median demonstrates an artifact close to point 5. It is common for homotopic medians to experience extra angles especially near sharper bends. This is not the case with the majority median.

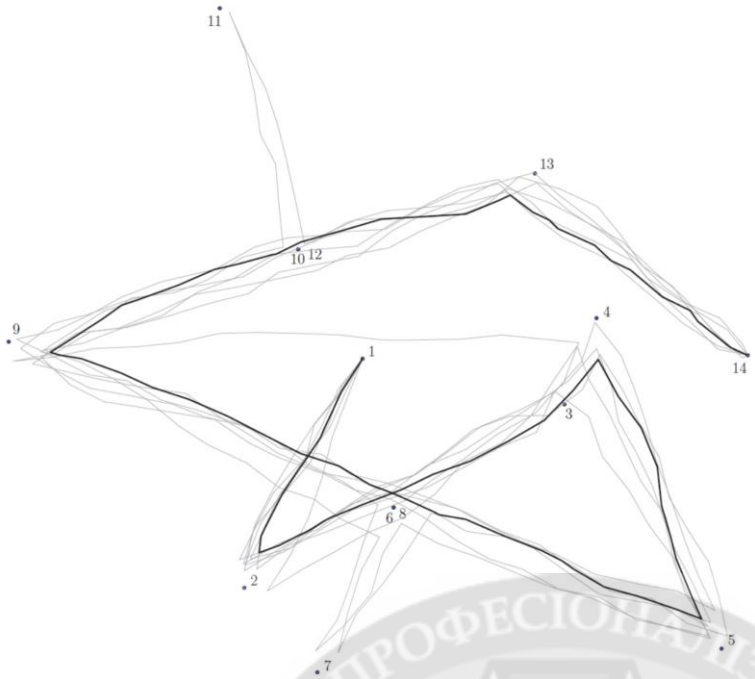

Figure 10 - The majority median trajectory (the black) for a bundle of input trajectories

The majority median satisfies most of the properties described in Section 1.4 for a representative trajectory. However, the median trajectory will not necessarily be locally centered since the focus is on the shortest path and not being in the center. In order to address this issue and improve the quality of the computed median, an alternative version of the majority median algorithm is also presented in [4]. The improved version differs only in assigned weights for edges when Dijkstra's algorithm is applied. In the original algorithm, the weight of an edge is just its Euclidean length. In the improved version, the weight of an edge is its Euclidean length times the size of the buffer needed for that edge to become useful (i.e. to have at least $\frac{m}{2}$ trajectories intersect the buffer). As a result, the alternative version gives preference to edges that are in dense areas because they are cheaper, and thus it outputs a median that is often more in the middle of the input trajectories.

Fig. 11 shows an example of a bundle of seven input trajectories, which were generated synthetically with a variation probability of 20% [3]. The trajectory generator creates trajectories that deviate on a larger scale utilizing three types of global variations with a certain probability. These include 1) skipping a point in a sequence of points, 2) visiting an extra, random point between two consecutive points in a sequence, 3) visiting an extra point from halfway between two consecutive points, and then going back to the halfway position to continue following the sequence. The markers with no numbers attached are the additional points used due to the variations, and they are used by some trajectories only. For example, three out of seven trajectories use the marker halfway between points 2 and 3 to go up and back to the marker between points 5 and 6. The homotopic median misses points 2 and 3, because the faces in this part of the input were considered too small to place a special point (poles). This can be resolved by choosing special points in smaller faces too, but then fewer trajectories are homotopically equivalent which has other negative consequences. The majority median and its alternative have the right global shape, but the majority median does not stay in the middle when the trajectories are close, for example near point 7. The alternative majority median is slightly better, although it has artifacts in the dense region left of the middle.

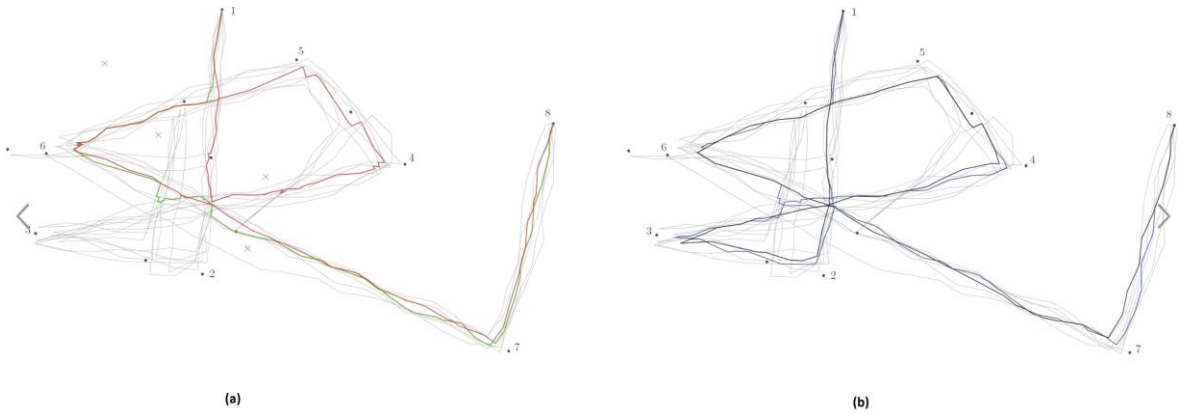

Figure 11 - Median trajectory for a bundle of seven input trajectories: (a) Simple median (green) and homotopic median (red), (b) Majority median (black) and alternative majority median (blue)

In general, the majority median and its alternative are supposed to slightly outperform the homotopic median. Nevertheless, the performance has to be defined based on the importance of the properties of the median. It is shown that the majority medians make fewer global errors (the correct shape of input trajectories are preserved) and have fewer artifacts, while the homotopic median is more in the middle when the trajectories are not deviating substantially [3]. There are several cases where the homotopic median performs better than the majority median [4]. An example in shown in Fig. 12. Although the majority median preserves and follows the overall path of input trajectories, it misses point number 4, because that area is already covered by another part of the median. On the other hand, the homotopic median is closer to point 4 and the result is better than the majority median in that part, even though it still misses the narrow space between points 2 and 3.

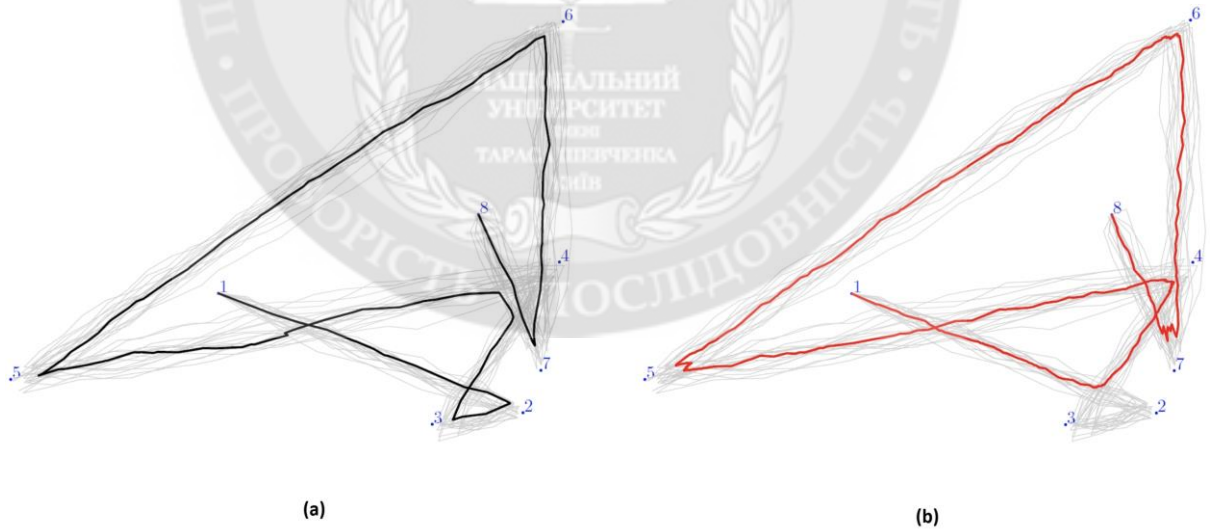

Figure 12 - The homotopic median trajectory outperforms the majority median: **(a)** Majority median misses' part of the path from point 3 to 4, **(b)** Homotopic median correctly follows the path from point 3 to 4

Conclusions. Most of the existing solutions to the problem of finding a middle trajectory do not take the temporal component into account. For some applications the time correspondence can not be clearly defined or it is logical just to ignore it. For example if we want to find a middle trajectory for several tourists taking a similar route. Due to different preferences, we can not use the

temporal information directly as they can take different paths on different times. If we include the temporal component, then we might find a completely different trajectory that is not irrelevant and pointless. Even following the same route, it can be that they have to stop at different locations. In this application, the temporal information is not relevant and can be simply ignored. However, in another application, it can be quite different. Suppose, we are not only interested in the route taken by most of the tourists but also the time it takes to complete an ordinary walking tour of a district. This can be difficult to compute because we have to ignore the stop incidents (visit tourist attraction, rest, eat, taking photos, etc.), which can be at different locations or can have different lengths. In this case we have to find a way to normalize the routes with regards to the timing information.

The methods that deal with temporal information of middle trajectories fall into two categories: 1) methods that are based on the notion of equal time slices, in which a middle trajectory is computed by finding a middle (representative) point for each time slice, 2) methods that assign timing data as a post processing step to a middle trajectory.

In addition to the properties described in Section 1.4, using temporal information for a representative trajectory, additional properties may be considered. These properties shall utilize the timing data in a way, e.g., the duration and speed of a middle trajectory should be comparable to the input trajectories in a bundle. The calculations of these properties is not straightforward and can be troublesome. One might define the duration of a trajectory as the time difference between the first and the last data points taking into account the stay points, or opt to exclude them from the calculation. Measuring the speed of a trajectory can also be done using the minimum, maximum or average speed, though it can be meaningless and not representative at all if several different scenarios and obstacles exist on the path.

The way the speed is determined for a middle trajectory influences its duration as well. If the calculation of the speed is based on the speed(s) of the input bundle, then the total duration of the middle trajectory can be a lot lower (similar speeds in combination with a shorter distance, which normally happens for the middle trajectory). A drawback would be that the middle trajectory can not be used to determine at what time a certain location will be visited. On the other hand, if the calculation is based on the total duration of the middle trajectory, then the middle trajectory can not be used to determine the speed in between the points. Hence, in this case, the middle trajectory should only be seen as a series of points where the time stamps represent the time at which the location will be visited, without having any information for what happens between two points. Regardless of the method that is used, we can set an upper bound on the speed that is based on the highest speed in the input. If this bound is exceeded, the result is most likely not a realistic representation of the input and either needs to be refined or rejected as a usable result. An example of such a scenario causing a trajectory to exceed speed bounds is when a trajectory spans great distances in a really short time [2].

Assuming that similar moving objects following the same itinerary behave in a similar way and move along an optimized main route (middle trajectory), Etienne et al. [5] proposed an approach to analyse the trajectories of these objects in order to infer several spatio-temporal patterns and then, to qualify their behavior by comparing their trajectories to these patterns. They first present a method to extract and filter trajectories of moving objects following a similar itinerary. This is done in several steps:

1. Formalizing the concepts of zones and itineraries, trajectories of same type of objects moving along the same path of an itinerary are extracted (a set of homogeneous bundle of trajectories).

2. Trajectories with an important gap between two consecutive positions or erroneous positions are filtered from the bundle in order to improve subsequent statistical analysis.

3. Filtering out starting and ending positions of trajectories within the departure and arrival zones in order to compute trajectories for which departure and arrival positions are independent from time of transmission (spatial shifting of trajectories). Without this filtering, measurements can be biased in the spatio-temporal patterns defined later.

4. Indexing and simplifying trajectories using a spatio-temporal variant of Douglas & Peucker filter to optimize subsequent computations by retaining only significant positions of trajectories while keeping information about both speed and heading changes.

5. Computing a relative timestamp for each position of a trajectory to ease distance and time comparison between trajectories. Timestamps of positions are useful to compute speed and order each position within a trajectory. The relative timestamp for each position of a trajectory indicates the time duration since the starting position of the trajectory.

6. Normalizing the timestamps of all the trajectories of a bundle to avoid spatial distortions introduced by slightly different speeds of moving objects. To compute this relative normalized timestamps, first of all, the median duration (D_m) of the bundle is calculated. Using this duration, a normalization process is applied to all positions so that each trajectory in the bundle begins at a time 0 and ends at (D_m).

After the initial phase of selecting and extracting the similar trajectories in a homogeneous bundle, the middle trajectory is computed by statistical analysis [6]. Considering the normalized timestamps previously, for each position of each trajectory in the bundle, positions of other trajectories are interpolated using their normalized timestamps. Then, at each normalized timestamp, a median position is selected using the median value of coordinates (latitudes and longitudes) of each position subset [7]. This creates a subset of positions for the whole trajectories in the bundle. Note that only meaningful positions that were selected by the spatio-temporal Douglas & Peucker algorithm are considered, so that the computation process is only applied on sub-parts of trajectories where mobile object behavior changes. Subsequently, the computed median positions are ordered according to their normalized time to create the middle trajectory for the bundle. Finally, this middle trajectory is also filtered using the spatio-temporal Douglas & Peucker algorithm. Listing 1 presents an outline of the algorithm for the computation of the middle trajectory.

```
for each trajectory Tr of the HGT_AIT do
    Delete erroneous trajectories
    Spatial shifting of starting and ending positions
    Douglas_Peucker_ST(Trajectory Tr)
    Temporal normalization using median duration t_m
end for
Algorithm Main_Route_Computation(HGT_AIT)
for each trajectory Tr_i of the HGT_AIT do
    for each position P_i of Tr_i do
        Let tn_i be the normalized time of P_i
        for each other trajectoiries Tr_j of the HGT_AIT do
            Interpolate the positions P_j at normalized time tn_i
            Add P_j to the subset of positions EP_i
        end for
        Compute median position P_med of EP_i
        Add P_med to the main route R_IT at normalized time t_ni
    end for
end for
return Douglas_Peucker_ST(Trajectory R_IT)
```

Listing 1 - The outline of the algorithm described in [5] for computing the middle trajectory in a bundle using the temporal information

Fig. 13 (a) depicts an example of a homogeneous trajectory bundle composed of 506 trajectories plotted in black. Looking at the figure shows that same-type moving objects with the same itinerary globally follow a main route (the middle trajectory) [8]. The cloud of dark dots shown in Fig. 13 (b) represents the subset of positions at a normalized timestamp, the large white dot indicates the median position of the whole subset. All these median positions ordered by theirs normalized timestamps compose the middle trajectory plotted in white in Fig. 13(a).

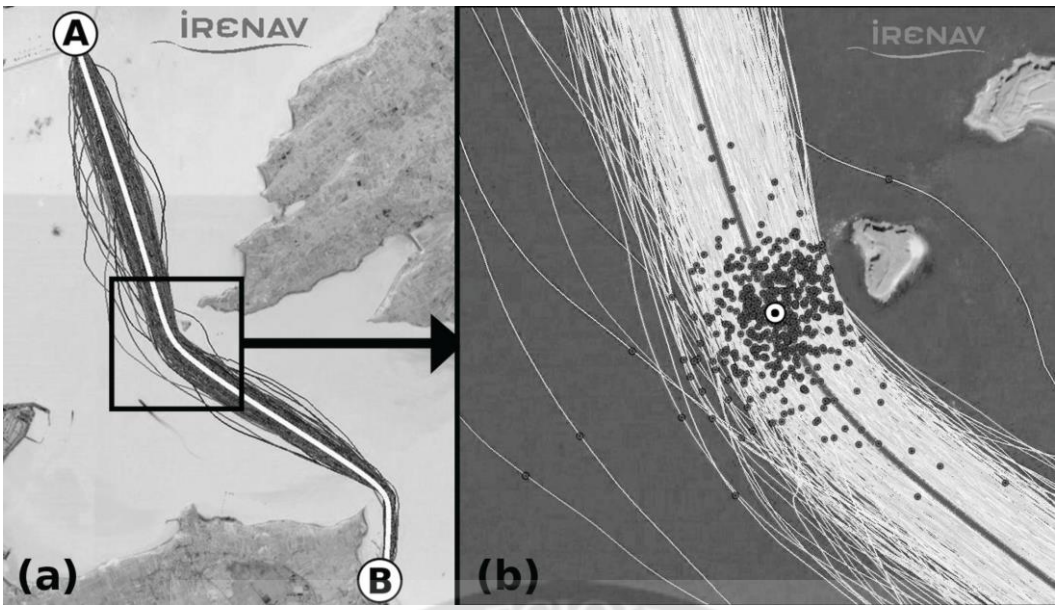

Figure 13 - A homogeneous bundle of trajectories and the computed middle trajectory

The main computational part of the algorithm described in listing 1 does not seem to the most efficient way of solving this problem in practice, however the asymptotic time complexity of the algorithm seems to be intact for more efficient alternative implementations. Another issue would be that for real world data, it is unlikely to identify exact locations known for all (equal) timestamps. For each input trajectory, in the worst case, there are $O(nm)$ data points to process because there are n timestamps and m trajectories, each of which might contain unique timestamps. This gives for a total running time of $O(nmT(m))$ for the main part of the algorithm, where $T(m)$ is the time it takes to process m points of equal time.

REFERENCES:

1. Buchin, K., Buchin, M., van Kreveld, M., Löffler, M., Silveira, R., Wenk, C., Wiratma, L. (July 2013), Median Trajectories**,** *Algorithmica* 66(3), pp. 595–614.

2. Vermeulen, T. (October 2013) Algorithms for finding a middle trajectory, Master's Thesis, TU Eindhoven.

3. Van Kreveld, M., Wiratma, L. (November 2011), Median trajectories using well-visited regions and shortest paths, In Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 241-250.

4. Wiratma, L. (2010), Following the Majority: A New Algorithm for Computing a Median Trajectory, Master's Thesis, Department of Information and Computing Sciences, Utrecht University.

5. Etienne, L., Devogele, T., Bouju, A. (2012), Spatio-temporal trajectory analysis of mobile objects following the same itinerary, In Advances in Geo-Spatial Information Science, 10, pp. 47–57.

6. Biagioni, J., Eriksson, J. (2012) Inferring road maps from global positioning system traces: Survey and comparative evaluation. Transportation Research Record: Journalofthe Transportation Research Board 2291, pp. 61-71.

7. Y.Chen, J. Krumm (2010) "Probabilistic modeling of traffic lanes from GPS traces", Proceeding soft the 18th SIGSPATIAL International Conference on Advancesin Geographic Information Systems, ACM, pp. 81-88.

8. B. Xu, M. Stroila, J. MacFarlane (2014) "A Before and After Study for Travel Time of Unconventional Intersections Using Probe Data", Proceedings of ACMSIGSPATIAL International Workshopon Computational Transportation Science.

д.т.н., проф. Заславський В.А., Пушкаренко Ю.В.

АНАЛІЗ ТРАЕКТОРІЇ ГОЛОВНОЇ КРИВОЇ

Збільшення доступності джерел даних геолокацій дозволяє використовувати їх в процесі автоматизації створення карт, корегувати зміст існуючих карт, а також надає потенціал для створення джерел істини (тобто ground truth) як у військовому, так і в цивільному аспектах. Це необхідно для підтвердження якості існуючих карт, а також для їх публікації з використанням різних джерел для досягнення комплексного результату створення мап надвисокої чіткості. Основна ідея полягає в тому, щоб формувати кінцеву карту не тільки базуючись на супутникових знімках, а й використовувати інші джерела і в тому числі дані геолокацій. Ми не заглиблюватимемось в сам процес формування карт, а зосереджуємось на проблематиці зіставлення карти з точки зору траєкторій отриманих з GPS, оскільки власне сам процес отримання за своєю природою є дуже спотворений завадами. В даній роботі здійснюється аналіз методів, заснованих на формуванні траєкторії головної кривої, сформованої з "сирих" даних GPS-локацій. Яскравим прикладом може бути геолокація яка отримана з телефону всередині автомобіля та є «зашумленою», тобто локація не обов'язково збігається з фактичним географічним положенням автомобіля у певний момент часу. Якщо припустити, що автомобіль їде вулицею згідно з правилами, то такі дані геопозицій на великій виборці можна зіставити з розташуванням вулиць за допомогою алгоритмів зіставлення карт. Використання серії покращень на великій виборці такими алгоритмами стабілізує і узгоджує положення об'єктів на карті, особливо коли дані локацій «спотворені завадами» або неповні, такими прикладами є різнотипність доріг, розташованість або близькість одна до одної (перехрестя, мости, тунелі, з'їзди).

Одним із таких підходів та пов'язані з ним алгоритми - є аналіз траєкторії головної кривої (або компоненти). Аналіз траєкторій головних кривих останнім часом привертає значну увагу завдяки технологічним досягненням у навігаційних і картографічних системах. Тим не менш, деяким фундаментальним концепціям все ще бракує ретельного вивчення. Ідентифікація середньої (репрезентативної) траєкторії в низці траєкторій є такою фундаментальною проблемою. Якщо не заглиблюватись у сутність судження, середня траєкторія - це траєкторія, яка лежить всередині сукупності траєкторій. Втім, дане твердження є далеким від вичерпності.

Ця дослідницька робота зосереджена на концепції знаходження траєкторії головної кривої серед низки траєкторій із конкретними точками відправлення та призначення. Основна ідея полягає у використанні інформації про час, пов'язаної з траєкторіями, для покращення існуючих методів аналізу траєкторій. Ми досліджуємо концепцію головної кривої, пов'язану з інформацією про час, також даємо огляд алгоритмів щодо всіх існуючих методів, аналізуємо час роботи в найгіршому випадку та показуємо, що за певних припущень такі методи, як хронометраж, можуть бути реалізовані ефективно.

Ключові слова: Узгодження карт, геопросторовий аналіз, обчислювальна геометрія, кластерний аналіз, дані геопозицій, виокремлення геометрії доріг, дорожні карти, аналіз просторових даних.