Window Operators for Processing Spatio-Temporal Data Streams on Unmanned Vehicles

Tobias Werner and Thomas Brinkhoff

Jade University of Applied Sciences, Institute for Applied Photogrammetry and Geoinformatics, Oldenburg, Germany,
{tobias.werner, thomas.brinkhoff}@jade-hs.de

Abstract. Unmanned aerial and submersible vehicles are used in an increasing number of applications especially for data collection in misanthropic environments. During a mission, such vehicles generate multiple spatio-temporal data streams suitable to be processed by data stream management systems (DSMS). The main approach of a DSMS is limiting the elements of a stream by using sliding and tilting windows with time intervals as temporal condition. However, due to varying vehicle speed and limited on-board resources, such temporal windows do not provide adequate support for spatio-temporal problems. For solving this problem, we propose a set of six new spatio-temporal window operators in this paper. This set comprises of sliding distance, tilting distance, tilting waypoint, session distance, jumping distance and an area window to limit stream elements based on spatial conditions. Each of the listed operators provides an individual behaviour to support sophisticated applications like spatial interpolation and forecasting. An evaluation based on an example trajectory shows the benefit of the presented operators for spatio-temporal applications.

Keywords: Spatio-Temporal, Data Stream, Window Operator, Moving Object, Unmanned Vehicle

1 Introduction

The use of autonomous vehicles performing observation tasks has increased in recent years. Unmanned aerial vehicles (UAV), autonomous surface vehicles (ASV) and autonomous underwater vehicles (AUV) or similar vehicle types are typically equipped with multiple sensors in order to observe environmental phenomena and to search for features of interest [1]. Those vehicles are often limited in resources. In many cases, the distribution of data between vehicles in a network is restricted or impossible due to a low bandwidth, missing signal strength or low energy level. Especially submersible vehicles are subjected to the limits of an acoustic based underwater communication system, which offers a maximum transmission rate of a few kilobytes per second [2]. In addition, their small power

AGILE: GIScience Series, 1, 2020.
Full paper Proceedings of the 23rd AGILE Conference on Geographic Information Science, 2020.
Editors: Panagiotis Partsinevelos, Phaedon Kyriakidis, and Marinos Kavouras
This contribution underwent peer review based on a full paper submission.
https://doi.org/10.5194/agile-giss-1-21-2020 | © Authors 2020. CC BY 4.0 License.

supply restricts a participation on long-during missions and prevents using powerful hardware for processing sophisticated data. Nevertheless, each sensor on an UAV, ASV or AUV generates an unbound amount of data elements that are associated with observation time and vehicle location. This information is the base for planning paths [3], preventing collisions, controlling the vehicle and supporting mission plans [4]. Therefore, it needs to be evaluated by the on-board system in situ.

One example is the search for the source of a contamination [1]. The mission time for observing indicating phenomena depends on the available vehicle resources. Especially the on-board power supply is typically very limited. In some cases, the vehicle is not moving or is moving slowly. The slower the vehicle drives the more observations are located at almost the same location. This spatially redundancy data burdens the system unnecessarily and should be reduced to save resources.

In regard to the previously described task of searching an arbitrary source, the on-board mission plan can be supported by estimating the trend of past observations as well. If those observations are used to generate gradients, it is possible to determine whether the vehicle is moving away from or approaching the source.

Another example is the observation of features of interest in a certain area. The vehicle has to take into account the observations that are within a predefined mission area. Elements that are located outside, do not need to be processed by the vehicle. Discarding insignificant data leads to a reduced processing load while the vehicle is maneuvering to the mission area or is coming back to its home station.

One approach for solving the described issues is to use a conventional spatial database management system (DBMS) for organisation and querying of data. However, traditional spatial database systems are focused on dealing with low dynamical spatial data instead of managing a continuously changing database [5]. Moreover, traditional spatial database systems focus on working in a network-based multi-user environment and are not designed to work with limited resources and a single peer.

An alternative approach is the use of a data stream management system (DSMS) that is specialized to operate on a continuously changing data [6][7][8]. In contrast to a conventional database system a DSMS defines a query language, entities, operators and execution plans for processing data streams. One key feature is the introduction of windows which limit elements of an unbounded stream by measuring the elapsed time (time-based window) or by counting elements (row-based window) [9]. But, those windows just consider time dimension and are often not effective on spatial and spatio-temporal issues. Especially, in the case of vehicles with changing speed, time-based or row-based windows are inefficient.

To handle the described issues, we see the need of specialised windows that take into account temporal dimension and spatial dimension. Therefore, we designed and implemented six different spatio-temporal window operators. These operators use spatial and spatio-temporal conditions to reduce the number of elements in a data stream. The design of the operators was motivated by the requirements of autonomous underwater vehicles.

This paper is divided into six sections. The next section presents related work about processing data streams. The third section gives an overview on temporal windows and illustrates their properties in detail. Afterwards, spatio-temporal window operators and their principles are introduced. To illustrate the benefits of the presented operators, Section 5 applies such windows on an example vehicle trajectory and demonstrates their basic properties and advantages compared to time-based windows. The last section provides a summary and gives an outlook to future work.

2 Related work

Unmanned vehicles generate a mass of high dynamic spatial data. The arise of autonomous vehicles leads to new data sources that can help to improve efficiency in several tasks like environmental monitoring. However, such spatio-temporal data sets are complex and need adequate processing techniques [10]. Typical DBMS are not prepared for dealing with such data because they are developed for managing and querying spatial data that are static or offer low dynamics. Especially, indexing of rapid changing data is a challenge [5]. Highly specialized systems are developed for dealing with vast amount of complex spatio-temporal data. The disadvantage of such systems is that they depend on sophisticated infrastructures [11] which often do not exist on unmanned vehicles. A more suitable approach for processing streams on-board is the use of a DSMS [6][7][8]. In the early years, major issues of such systems were the limited memory and the use of blocking operators on continuous streams [12]. Current systems are able to process continuous, rapid, time-varying and boundless data. However, computing exact results based on recent data is still non-trivial. As a result, several approaches were made that use approximations to compute sufficient query results. One popular approach limits elements in a stream based on their temporal properties by windows [9][13][14]. A window uses an interval for validating each stream element. As soon as the condition is not satisfied anymore, the element is discarded. These windows are well supported by modern DSMS. PipelineDB [15] Apache Flink [16] and Odysseus [17] are examples.

Windows are also used in a spatio-temporal context: They were applied in social media to analyse and organize geo-tagged streaming text messages [18]. The medical domain uses sliding windows to process only recent sensor data, for predicting the upcoming fall of a person [19] as example. DSMS that are specialized for vehicles make use of such windows as well. As main component of an automotive, the DSMS provides a central node for distributing and analysing sensor data [20].

We developed an architecture for processing spatio-temporal data streams on AUVs [21]. The advantage of our system is that it is specialized to operate on resource limited vehicles and is developed as an extension for the SQLite database engine. This approach allows operating on traditional tables and data streams together in one environment. Our architecture includes time-based and row-based windows. The disadvantage of such operators is that they do not consider the special needs of moving objects. Some use cases are satisfiable by using pure temporal windows. As soon as spatial conditions do exist, the operators fail or provide a massive overhead.

3 Temporal windows

This section reviews existing temporal window operators. In the following of the paper, we adopt their main principles and reuse them in a spatio-temporal context.

Several types windows can be used to limit the elements of a infinite data stream [9]. Figure 1 compares the behaviour of four temporal window types. The first row with label *data* contains the incoming elements of a data stream while the coloured vertical lines at t_1, t_2 and t_3 mark the instant of time when a window is queried. For each window type, the result set is depicted by using a rectangle. The rectangles are as colored as their corresponding time of querying. The first row of a window type depicts the result set at t_1, the second at t_2 and finally the third at t_3.

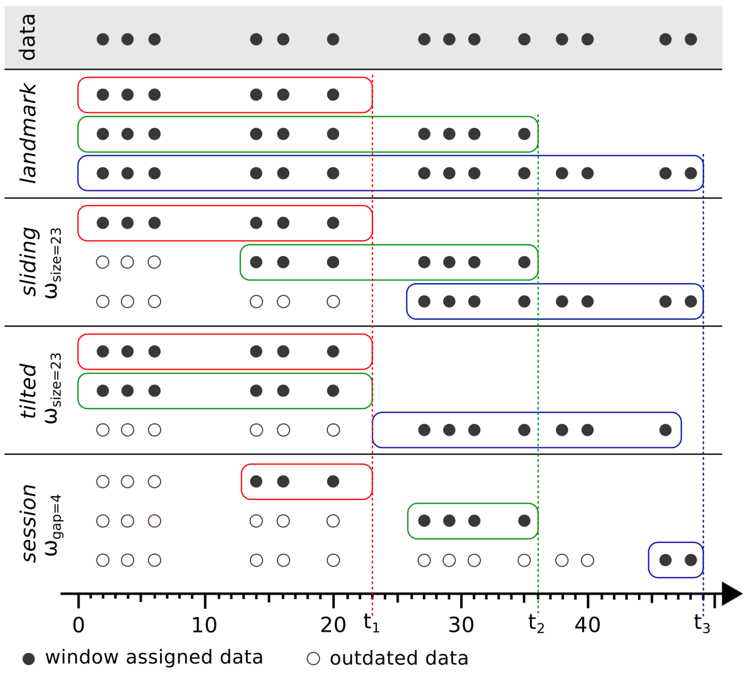

Fig. 1. Temporal window types. Modified illustration according to [9] and [13]

The *landmark window* (also known as global window) holds all elements since the beginning of the stream and returns everything seen so far at the time of querying. This operator needs to be used when all elements of a stream are significant and every element shall be processed. The *sliding window* holds elements that are not older than a specified age. The time interval for the validation process is defined by the window size ω_{size}. Updates on the result set are done continuously and identical elements in multiple result sets are typical. This operator has to be used when an application needs a latest snapshot of past data. The *tilted* window (also known as tumbling [14] or fixed window [13]) is defined by a window size ω_{size}, too, but does not update the assigned result set continuously. As soon as a window period is completely in the past, it becomes the new active window instance which holds the result set. The tilted window operator has to be used when a sophisticated application needs a more static snapshot of past data without redundancy. The *session window* provides elements that are close together in time. As soon as the temporal gap ω_{size} between elements of the stream is bigger than specified, a new result is becoming the active instance. This window can be used for applications that depend on processing groups of elements.

4 Spatio-Temporal windows

In this section, we introduce six operators for solving the described issues and discuss their benefit compared to pure temporal windows based on an exemplary mission.

Our basic idea is to apply the underlying principles of the temporal window types to the spatial domain. Thereby, the process of assigning elements to a window instance is using spatial conditions instead of using temporal conditions. For that, there is a need for strictly ordered elements. Temporal windows presuppose a temporal order of incoming data. Elements of a stream can have multiple explicit and implicit temporal properties which can be used for ordering. Implicit information can be added by the DSMS to hold the time of element generation to support internal processing. Explicit properties typically are dedicated attributes that are set by externally. Only elements that are newer than the last seen are processed. Otherwise, their ordering is wrong and they are discarded. We reuse this temporal information of each element to define the trajectory of a moving object.

4.1 Exemplary mission

To show the issues of temporal window operators, we use the scenario depicted in Figure 2. This scenario is a simulated movement of an AUV for an observation mission. This data set is also used for evaluating the introduced operators later. The simulated vehicle estimates its location in a frequency of 10 Hz and uses a steady propulsion that can reach a speed of 2 meters per second. Due to several effects, the density of recorded locations changes over time. The slower the vehicle is moving the higher is the density of locations and vice versa. This density is important because it mostly corresponds to the concentration of observations in the proximity.

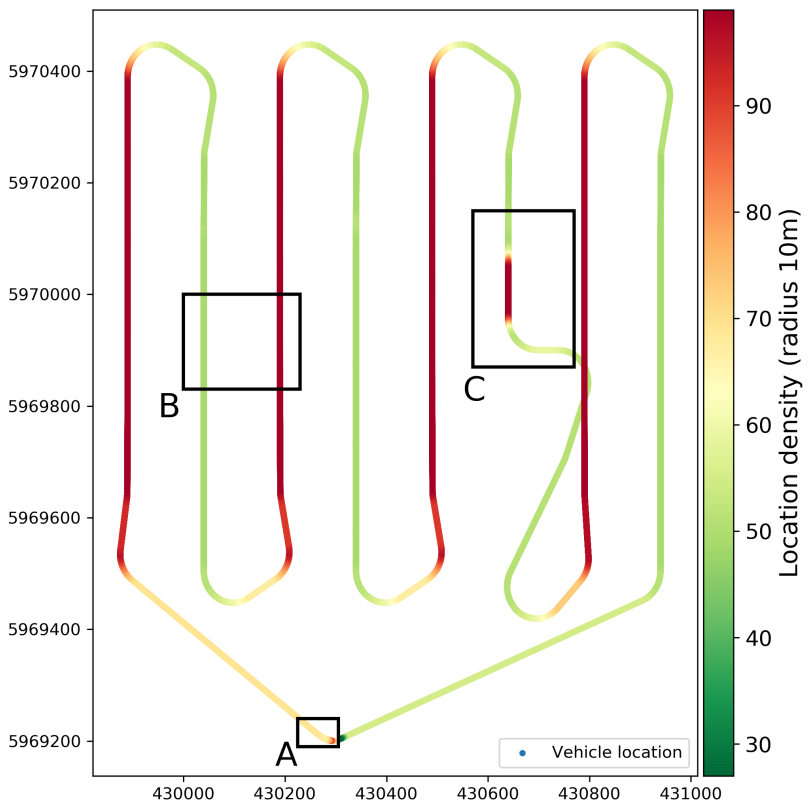

Fig. 2. Simulated trajectory of an AUV during an observation mission

At some point, the mission of a unmanned vehicle starts (see rectangle A in Figure 2). Typically it is not moving yet. As soon as hardware check-ups and calibration processes have been finished, the vehicle starts moving and accelerates.

The speed of an unmanned vehicle can be heavily influenced by environmental forces. UAVs can be accelerated or slowed down by gusts. The actual speed of USVs and AUVs depends on water currents. Like in area B of Figure 2, the direction of the vehicle movement and the direction of water current lead to a higher density of locations if they are directed in the same direction or to a lower density of locations if they are in opposite directions. The more powerful the water current is, the higher is the expected influence on the density of locations.

The online mission plan of autonomous vehicles has to react on unpredictable situations like obstacles in situ. Area C in Figure 2 shows a slow-down by the vehicle for calculating alternative routes around an obstacle. Depending on the system infrastructure, this step can be relatively time-intense before the vehicle accelerates to fulfill the mission.

Temporal windows from Figure 1 can be used to limit elements of a stream based on spatial condition as long as the vehicle speed is constant. Getting a snapshot of observations that are within a travelled distance of 100 m as example. Assuming a constant vehicle speed of three knots (1,54333 m/s), a sliding or tilting window with at least $\omega_{size} = 65sec$ is needed. However, a constant vehicle speed can not be guaranteed. As soon as the vehicle accelerates or gets slower, the corresponding window size has to be adjusted. In the worst case, this adjustment has to be performed on every observation. This recurring process is additional work and can be avoided by using one of the following operators.

4.2 Sliding distance window

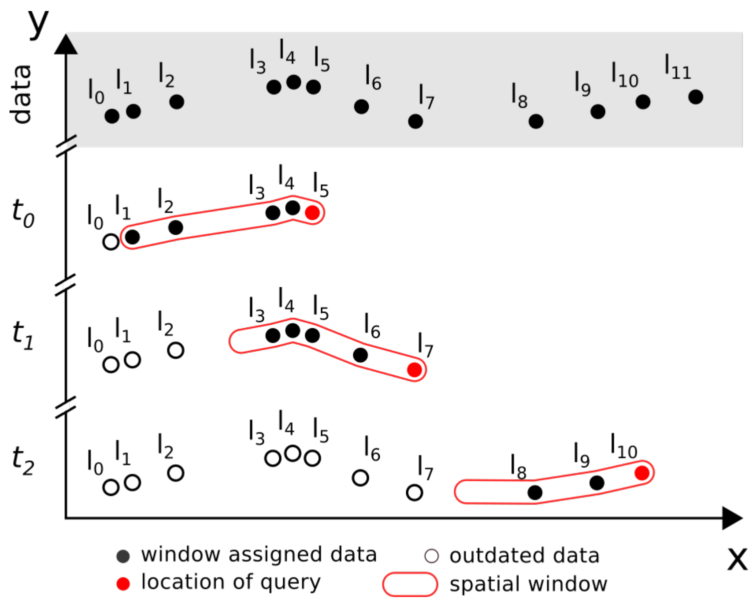

Fig. 3. Sliding distance window

The sliding distance window is inspired by the temporal sliding window, but it uses the travelled distance as window condition instead of a time interval. The principle is shown in Figure 3 based on an example. The waypoints l_0 to l_{11} are vehicle locations that have a temporal order which starts at l_0. t_0, t_1 and t_2

define consecutive points of time when the result set of the window is queried. The red point indicates the current vehicle location. Elements that are assigned to the window instance (and fulfill the window condition) are filled while the discarded elements are not filled.

Time-based windows $\mathscr{W}_\omega^{time}$ are defined in formula 1 [22]. They limit the elements of a stream $\mathscr{S}$ by a temporal interval $\mathscr{T}_\omega$. The window ω defines the bounds of such interval and generates a finite set of data that is provided as relation $\mathscr{R}$.

$$\mathscr{W}_\omega^{time} : \mathscr{S} \times \mathscr{T}_\omega \rightarrow \mathscr{R} \tag{1}$$

For defining spatio-temporal windows $\mathscr{W}_\omega^{spatial}$, we adapt in formula 1 and introduce the spatial property $\mathscr{D}_\omega$ that specifies the condition for limiting the elements of a stream $\mathscr{S}$. In case of the sliding distance window, $\mathscr{D}_\omega$ represents the travelled distance. The result set $\mathscr{R}$ of a queried window instance contains every element that is located on the track and is within the window distance size ω.

$$\mathscr{W}_\omega^{spatial} : \mathscr{S} \times \mathscr{D}_\omega \rightarrow \mathscr{R} \tag{2}$$

For determining which elements are within the window, we define distance intervals and introduce the function δ in formula 3. This function calculates the distance on a trajectory from the beginning to the given location (or set of locations) whereas the last known location is defined as $\vec{l_n}$.

$$\mathscr{D}_\omega(\vec{l_n}) = \begin{cases} [0,\ \delta(\vec{l_n})], & if\ 0 \leq \delta(\vec{l_n}) \leq \omega \\ [\delta(\vec{l_n}) - \omega,\ \delta(\vec{l_n})], & if\ \delta(\vec{l_n}) > \omega \end{cases} \tag{3}$$

Consequently, every new generated location is assigned to the window instance instantly. Finally, as soon as an element is out of range it will be removed. In regard to Figure 2, a result set can be provided that represents observations on a specified length of the track without doing recurring adjustments of window parameters.

The property of the sliding mechanism assures that the data of the result set is always up-to-date. But, this feature depends on continuous updates on the result set, which can be massive due to data that is measured in high frequency. Sophisticated applications can be overstrained by processing all result sets. Furthermore, the sliding principle results in overlapping sets which cause redundancy in some cases. Applications, that bases on an incremental processing would not benefit by this type of window.

4.3 Tilting distance window

The principle of the tilting distance window is similar to the previously described sliding distance window and is specified by the parameter ω as well. But unlike the sliding approach, the tilting distance window does not change its result set continuously. Only as soon as the vehicle has travelled the specified distance, the assigned elements will change. An example is illustrated in Figure 4. Querying the window at t_0 provides the same result set as querying at t_1 because the travelled distance since the last window instance is not larger than the specified distance. This causes a skip of the blue-framed elements l_6 and l_7. At t_2 the window instance tilts and changes the assigned elements. According to Figure 2, this operator generates result sets which represent observations in a certain travelled distance.

Formally, this operator is defined in formula 4, which makes use of the gaussian floor notation.

$$\mathscr{D}_\omega(\vec{l_n}) = \begin{cases} \varnothing, & if\ 0 \leq \delta(\vec{l_n}) \leq \omega \\ \left[\left(\left\lfloor \frac{\delta(\vec{l_n})}{\omega} \right\rfloor - 1\right) \cdot \omega, \left\lfloor \frac{\delta(\vec{l_n})}{\omega} \right\rfloor \cdot \omega\right], & if\ \delta(\vec{l_n}) > \omega \end{cases} \quad (4)$$

In contrast to the sliding window, the advantage of this window is a lower frequency of updates of the result set. Once a result set is defined, it will not get altered by adding or removing elements. As a result, sophisticated applications have to be triggered less often. But, a lower update rate leads to a skip of recent data and makes the results of an analysis more outdated or inaccurate. Applications that need input datasets without redundancy benefit from this type of window.

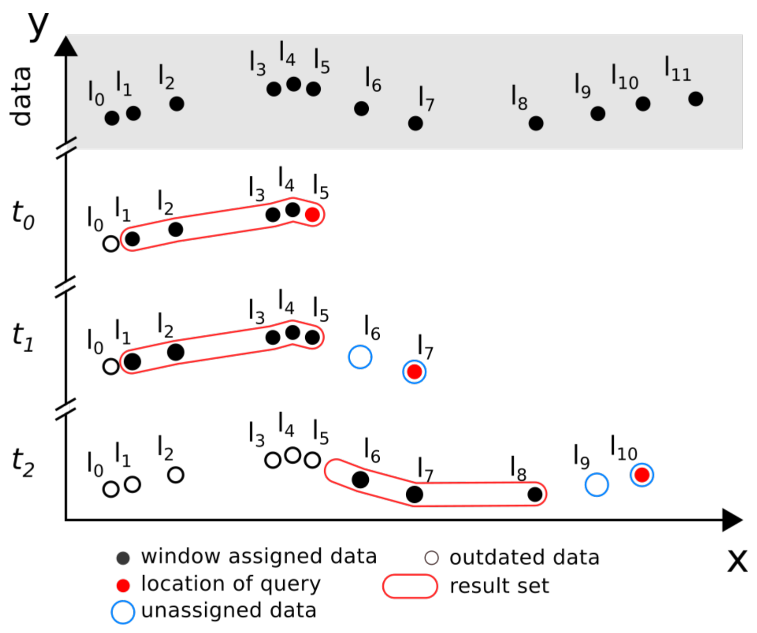

Fig. 4. Tilting distance window

4.4 Tilting waypoint window

The mission plan of self-driving vehicles often uses waypoints as markers to define the route to a target. In many cases, such waypoints are deliberately set and represent points of interest. The observation mission that is illustrated in Figure 2 uses waypoints to define the edges of the targeted route. For example, area B marks two edges that both were defined by two waypoints and have opposite directions. Estimating the trend of an observation along a predefined segment of a whole track cannot be solved by a window that is based on a steady distance or a steady time interval.

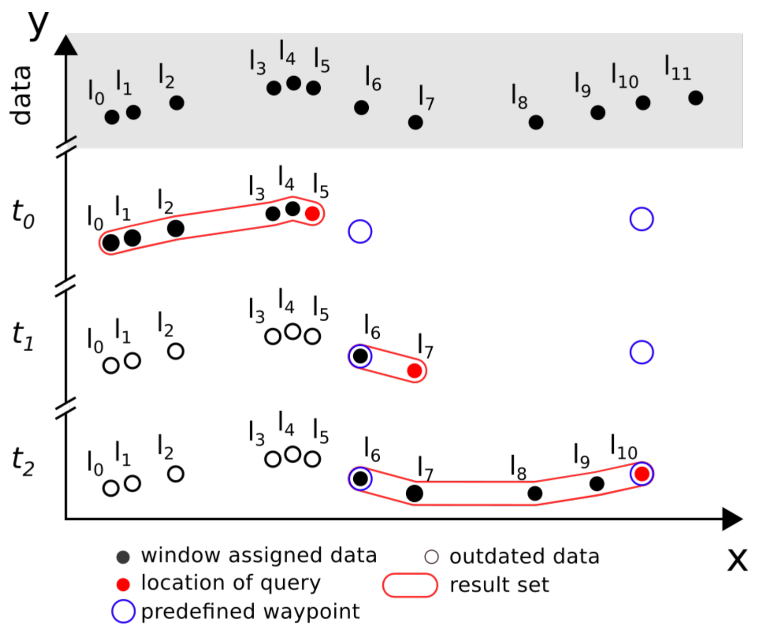

Fig. 5. Tilting waypoint window

As a solution, we provide the tilting waypoint window that is illustrated in Figure 5. This window declares points of interest as main condition for defining the event of tilting. This principle is very similar to the tilting distance window but allows using certain path segments as semantic breakpoints. As illustrated, all elements that are located in time between two predefined waypoints (including the waypoint itself) are assigned to the active window instance and thus are part of the result set. As soon as a waypoint is visited by the vehicle, like it does at t_1 on location l_6, the window will tilt. In contrast to the tilting distance window, the last element that is assigned to a window instance is the first element in the subsequent window instance. This double usage is helpful for representing segments as geometric edges. In respect to Figure 2, a processing of observations

https://doi.org/10.5194/agile-giss-1-21-2020 | © Authors 2020. CC BY 4.0 License.

can be based on specified track segments. Processing all segments that are not part of an evasive action (area C in the figure) is an example.

We formalize this operator in formula 5 and 6. κ is a set of locations for representing all waypoints and υ all visited waypoints. The max function selects the element with the highest value.

$$\upsilon = \{v \in \kappa \mid is\ visited\} \tag{5}$$

$$\mathscr{D}_\omega(\vec{l_n}) = \begin{cases} [\,0,\ \delta(\vec{l_n})\,], & if\ \upsilon\ is\ \varnothing \\ [\,max(\delta(\upsilon)),\ \delta(\vec{l_n})\,], & if\ \upsilon\ is\ not\ \varnothing \end{cases} \tag{6}$$

One advantage of the described window is that is allows a semantic approach for path segmentation. Thus, the calculation of trends for estimating phenomena based on classified segments is supported. One disadvantage is the movement accuracy that a vehicle offers. In the most cases, a waypoint is not visited exactly due to inaccuracies. Thus, the window does not tilt like expected. One approach for solving this issue is to approximate the waypoints by buffers.

4.5 Session distance window

The condition of the session distance window is based on a spatial gap ω_{gap}. As long as adjacent locations are within this gap, the associated elements will be assigned to the active window instance. As soon as a pair of locations provides a distance between two consecutive elements that is larger than the specified gap, a new window instance will be created which becomes the new active instance.

An example of the session distance window is illustrated in Figure 6. The location pairs (l_2, l_3) and (l_7, l_8) have a distance that is larger than ω. Therefore, window instances will be created as soon as the vehicle has reached l_3 and l_8. All following elements are assigned to the current window instance. The described behaviour arranges nearby elements into groups according to their distance from each other. In reference to Figure 2, this operator can group observations based on their distances and allows summarizing values based on their location. Like area A in the figure, where the vehicle is not moving or moving very slow. There are many elements that are very close to each other. Values that are within this small area can be grouped by the session distance window and summarized to calculate an approximation.

The session distance window is defined in formula 7 and 8. l is a set of all locations on the trajectory and l_i the i^{th} location on the trajectory.

$$\upsilon = \begin{cases} \varnothing, & if\ n \leq 1 \\ \{v \in l \mid \delta(\vec{l_i}) - \delta(\vec{l_{i-1}}) \geq \omega\}, & if\ n \geq 2 \end{cases} \tag{7}$$

$$\mathscr{D}_\omega(\vec{l_n}) = \begin{cases} [\,0,\ \delta(\vec{l_n})\,], & if\ \upsilon\ is\ \varnothing \\ [\,max(\delta(\upsilon)),\ \delta(\vec{l_n})\,], & if\ \upsilon\ is\ not\ \varnothing \end{cases} \tag{8}$$

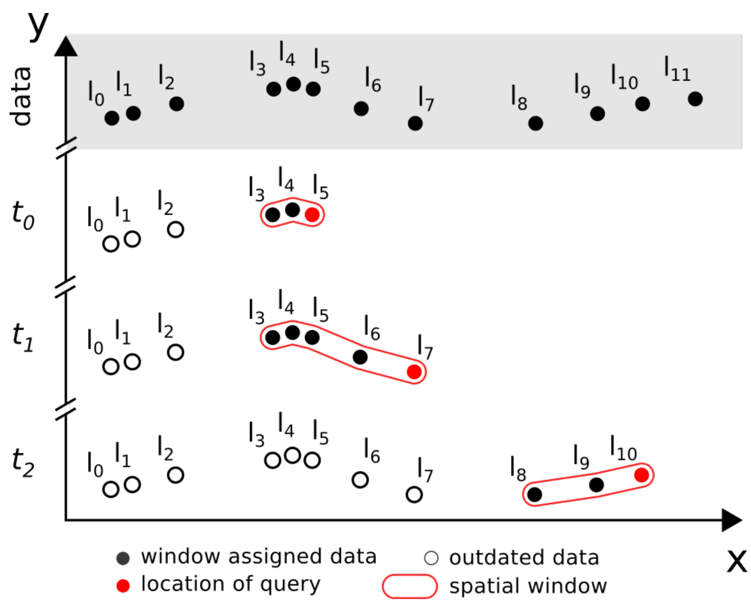

Fig. 6. Session distance window

Similar to the tilting distance window, the result sets share no elements together. Thus, no data redundancy is generated. However, the current active result set must be changed as soon as a new element has been assigned to that window.

4.6 Jumping distance window

Another principle of reducing the elements of an unbound data stream is sampling [9]. Such reduction approach is the basic idea of the jumping distance window. As soon as the associated vehicle has changed its location by a predefined distance ω, the next spatial element will be assigned to this window. Thus, a result set is provided that represents the environment in a lower resolution.

The principle of this window is illustrated in Figure 7. The first element that is assigned to the window instance is l_2. As soon as the vehicle has travelled a distance that is greater than ω, the next element will be assigned to the current result set. At time t_0 the next assigned element is l_5. This behaviour can lead to skip recent data. Like at time t_1 when the location is l_7 but l_6 and l_7 are not part of the result set. According to Figure 2, this operator can be used to select observations in a periodical distance. Especially for the processing of observations in area B where the density of locations is varying massively, this operator is helpful to provide observations in a uniform resolution.

The definition of this operator is given in formula 9. The operator *mod* calculates the rest of a division. ϵ defines a granularity and represents the accuracy of the spatial condition.

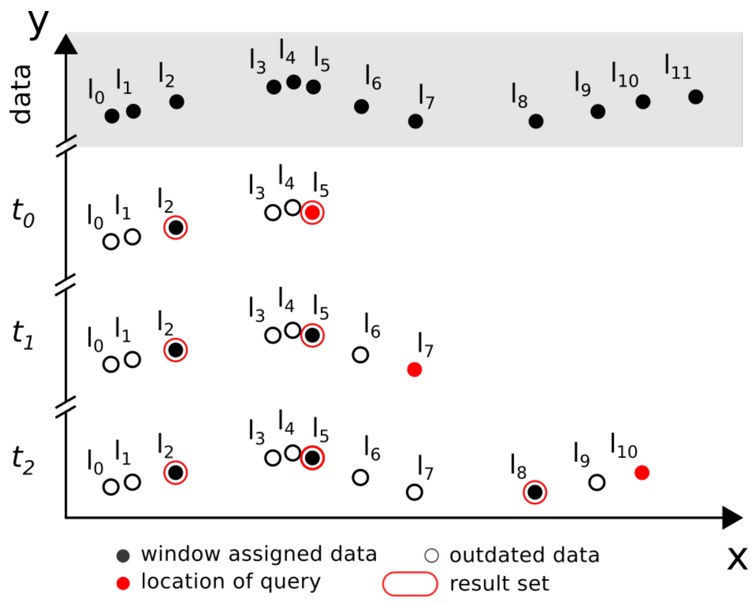

Fig. 7. Jumping distance window

$$\mathscr{D}_\omega(\vec{l_n}) = \{v \in l \mid \delta(\vec{l_i}) \mod \omega < \epsilon\} \tag{9}$$

The advantage of this window is that it can reduce the resolution of spatial data sets. This leads to a higher ratio of cost and benefit in sophisticated applications. Furthermore, recent data like observations at the current location are possibly ignored.

In a scenario where only a couple of recent observations are significant, this operator can be combined with other window types. After adding the sliding or tilting distance window, only recent elements will be added to the result set and the resolution will be reduced at the same time.

4.7 Area window

One main feature of a spatial database system is to provide efficient access on spatial data. The Dimensionally Extended 9-Intersection Model (DE-9IM) defines several spatial relationships that can be used in a spatial query. The main problem of spatial queries is that geometrical tests between geometries are a sophisticated task. One typical approach for speeding up such tests is using approximations. Minimal Bounding Rectangles (MBR) of exact geometries are mostly applied. Calculating the relation between two MBRs as prior filter step is fast and allows reducing the set of candidates massively. Because data stream processing systems are typically used in time-critical environments, the area window filters elements based on MBRs instead of complex geometries. In respect to the DE-9IM this window uses the topological *within* operator.

An example is shown in Figure 8. The red rectangle specifies the area which represents the spatial condition for assigning elements to the current window instance. Locations like l_3, l_4 and l_5 at the time t_0 are added to the result set while the outlying elements l_0, l_1 and l_2 are discarded immediately.

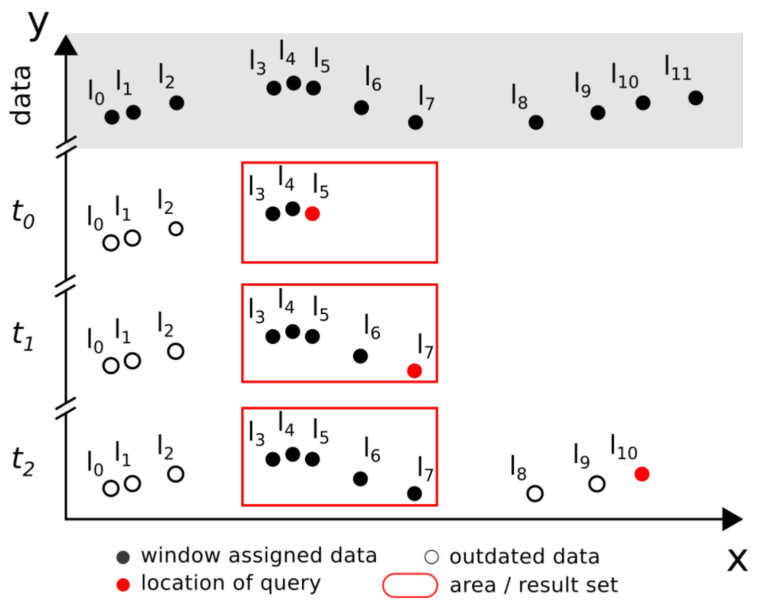

Fig. 8. Area window

Formally, this operator is described in formula 10. ω defines a bounding box that is bounded by the coordinates $(minx, miny)$ and $(maxx, maxy)$.

$$\mathscr{D}_{\omega}(\vec{l_n}) = \{v \in l \mid \omega_{minx} < l_{ix} < \omega_{maxx} \; and \; \omega_{miny} < l_{iy} < \omega_{maxy}\} \tag{10}$$

The advantage of this window is that only elements are processed that are within a specific area. Especially in a mission where the vehicle has to drive a long distance until it reaches the target area, the on-board system will benefit by ignoring data which is located in irrelevant areas. This situation can be observed in Figure 2. As soon as the mission begins, the vehicle starts on a location that is in area A. The vehicle now has to drive to the target area of the mission and later leave that area to come back to the starting point. Both parts of trajectory are completely insignificant and offer no observations that have to be processed.

This operator is suitable for a combination with other window types. A scenario which has a need for recent observations that are located in a certain area would benefit from adding a sliding distance window or tilting distance window. Even a jumping distance window can be added as third operator to reduce the generated result set.

5 Evaluation

The principles of the introduced spatio-temporal windows were presented in the previous section. This section evaluates them by using the trajectory that is presented at the beginning of this paper in Figure 2. We compare the amount of elements that the result sets include and describe their particularities. The sliding distance window is omitted due its similarity to the tilting distance window. They only differ in the time of updating the result set. The session distance window is not evaluated as well because the trajectory of our exemplary mission in Figure 2 represents a continuous movement. But, the session distance window is only reasonable to be applied in case of irregular variations in the distances between locations.

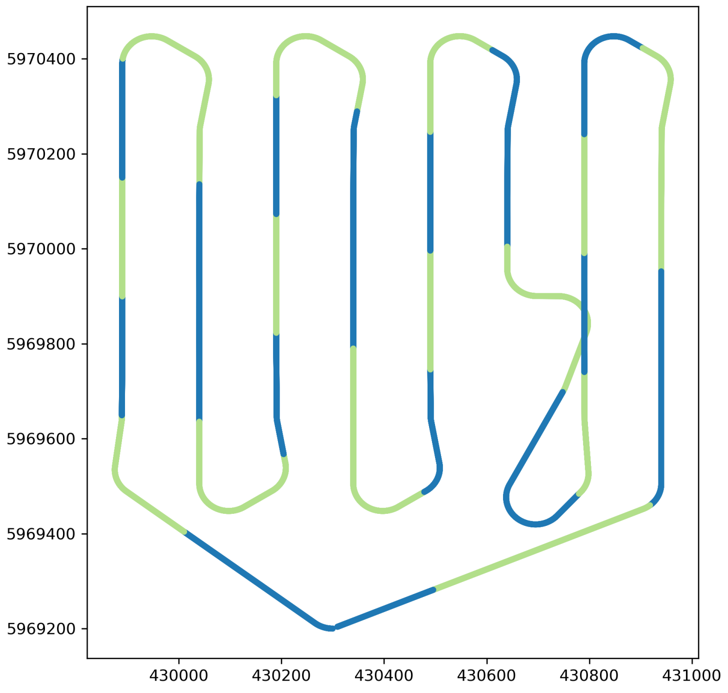

Fig. 9. Trajectory segments of tilting window with $\omega_{size} = 250sec$

5.1 Data and Software Availability

For illustration purposes, we implemented the window operators in PostgreSQL in combination with the spatial extension PostGIS. We also implemented such operators in our data management stream system that adds support to SQLite for processing spatio-temporal data streams by using the virtual table mechanism. Details about a technical implementation can be found here. Following the AGILE Reproducible Paper Guidelines, the software and all data supporting this publication is published at `https://figshare.com/s/cc758d056c8c6f193e52` and `https://figshare.com/s/3580e224f76adc0e3425` under the MIT-license.

5.2 Temporal tilting window

At first, we applied the conventional temporal tilting window with a size of ω_{size} of 250 seconds. The windows are illustrated in Figure 9 by alternating colours. Assuming a constant vehicle speed at three knots (1,54333 m/s) and a fixed observation frequency of 5 Hz, each window instance would provide a result set of 1250 elements that represent a track of about 386 meters. But, due to external forces and maneuvers the actual represented segment length for window instances is varying. This variation is illustrated in Figure 10. Each window instance represents an individual distance of the trajectory that reaches from 202 meters to 500 meters. Thus, retrieving a result set that covers a certain distance is not possible by a temporal tilting window.

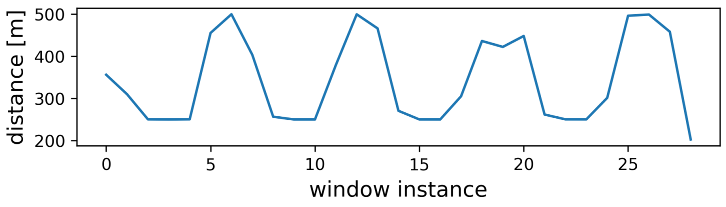

Fig. 10. Distances of tilting window with $\omega_{size} = 250sec$

5.3 Tilting distance window

The results of applying a tilting distance window with $\omega = 500m$ on the example track is illustrated in Figure 11. Each of the 20 window instances (except the last one) provides a result set that represents a segment of 500 meters. In contrast to the conventional tilting window, the associated time interval is not fixed but is correspondingly adapted. This adaptability is illustrated in Figure 12. The interval of each window depends on the actual speed of the vehicle and is between 249 seconds and 499 seconds. In other words, in order to guarantee the coverage of 500 meters we need a temporal window of (at least) 499 seconds. In contrast to the tilting distance, the average number of elements in that window would be higher by a factor of 1.4. Thus, the main advantage of this operator over the corresponding temporal window operator is that it guarantees the coverage of a given distance without the need to overestimate a corresponding period of time. This property also reduces the memory requirements significantly.

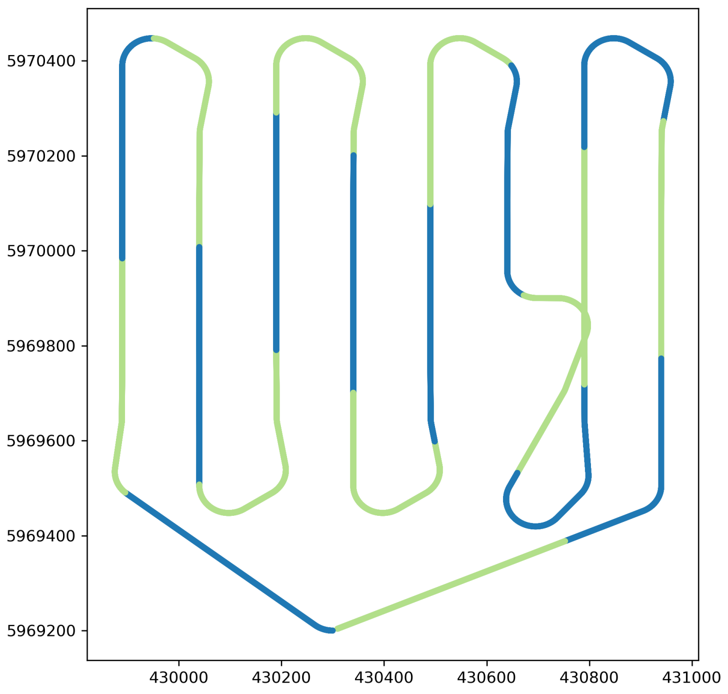

Fig. 11. Tilting distance window with $\omega = 500m$

5.4 Tilting waypoint window

Figure 13 shows the resulting window instances of the tilting waypoint window. 57 consecutive waypoints (depicted as red points) were generated by the mission plan for observing the targeted area with a resolution of 150 meters. The length of the illustrated window instances and the number of assigned elements depend heavily on the spatial distribution of the waypoints. Therefore, the length of the segments varies from 132 meters (378 assigned elements) up to 704 meters (1912 assigned elements).

The corresponding time intervals of the window instances and their adaptability to the actual situation are illustrated in Figure 14. Especially the first and

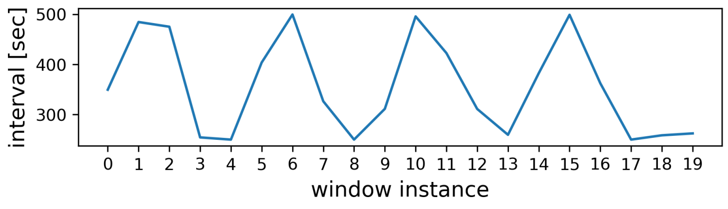

Fig. 12. Tilting distance window with $\omega = 500m$

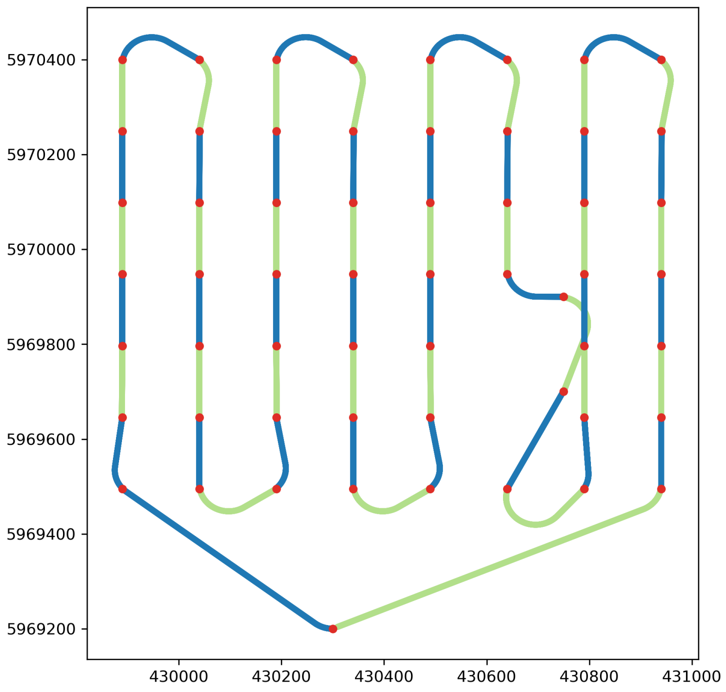

Fig. 13. Tilting waypoint window

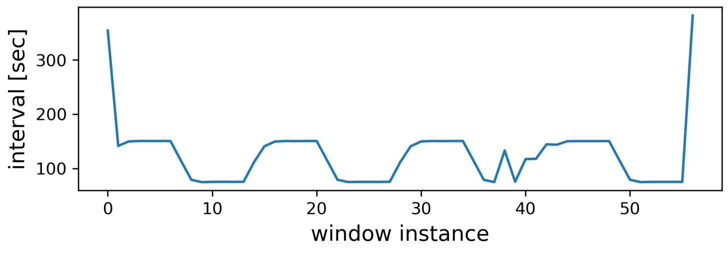

Fig. 14. Tilting waypoint window

last segment provide a relatively large time interval because they represent the segment of the trajectory where the vehicle drive towards the mission area and drive back to the beginning. As an advantage, this operator has the ability to split the trajectory into predefined segments by given points of interest. Again, we need only to process elements that fulfill a given spatial condition. A comparable space partitioning cannot be achieved by pure time-based or row-based windows.

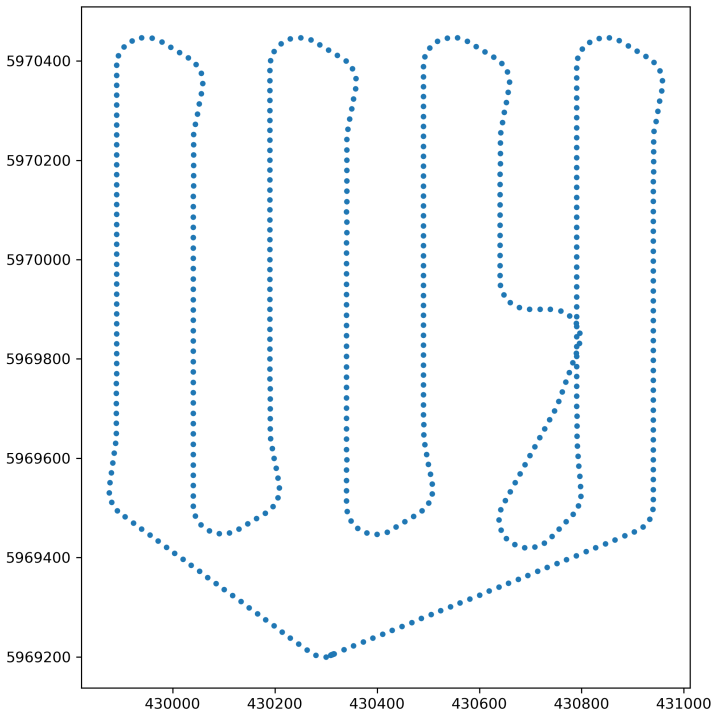

Fig. 15. Jumping distance window with $\omega = 20m$

5.5 Jumping distance window

Applying the jumping distance window for limiting the number of stream elements is illustrated in Figure 15. The depicted example uses a window with $\omega = 20m$ that provides one result set which is continuously extended over time. As a result, the complete trajectory of about 10 kilometres leads to 543 elements assigned to the window. This number corresponds to a reduction by a factor of 65 compared with the original data. In order to guarantee the same distance by using a temporal window, we would need a time interval of 17 seconds in minimum. Such an interval would lead to 419 result sets á 85 elements. The advantage of this window over the previously described operators is that a vast amount of observations can be reduced to a few observations based on their spatial distribution.

5.6 Area window

As described before, the area window uses a bounding geometry for limiting the incoming stream elements. In the illustrated example in Figure, 16 ω_{area} is set to the rectangle that defines the target area of the mission.

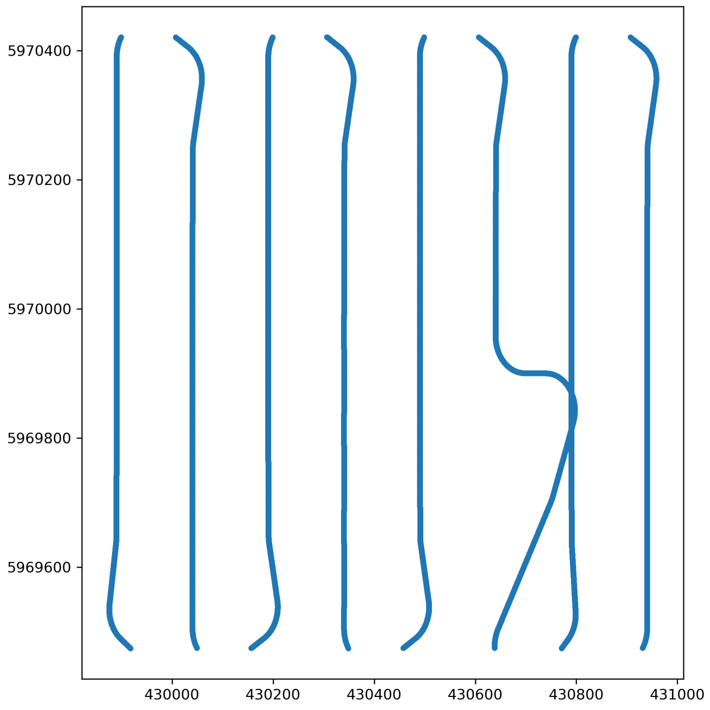

Fig. 16. Area window

If the vehicle only acts within the mission area, the result set will contain every produced stream element. Thus, the stream of elements can be theoretically boundless. The depicted trajectory consists of segments which are located outside the targeted area and thus are dismissed instantly. Based on the example, about 29 thousands of observations are finally part of the result set while 6 thousands are discarded. This behaviour allows focusing only on spatially significant observations. Furthermore, the area window is very suitable to be combined with other windows. Its combination with the jumping distance query of Section 5.3 reduces that result set by 27 % to 395 relevant elements.

6 Summary and Outlook

Unmanned aerial and submersible vehicles generate data streams that are spatio-temporal. Systems for processing such streams typically use temporal operators for limiting the incoming elements. Established window operators like the sliding and tilting window use time intervals as condition for validating and thus taking only the temporal aspect into account. But, during a mission moving, vehicles are faced with spatio-temporal challenges that need to be considered.

Table 1. Overview spatio-temporal window operators

Name	Parameter	Data type
Sliding distance window	ω describes a distance	double
Tilting distance window	ω describes a distance	double
Tilting waypoint window	ω describes a set of waypoints	point[]
Session distance window	ω describes a distance gap	double
Jumping distance window	ω describes a distance	double
Area window	ω describes an area	rectangle

In this paper, we presented the operators summarized in Table 1 for processing spatio-temporal data streams. Depending on the type, they use different parameters for defining the window condition. This comprises either a travelled distance, a predefined mission area or fixed waypoints. In addition, the behaviour of a window when it provides a new result set, depends on the window type as well. For showing the benefits of the operators, we applied them on an example vehicle trajectory.

The introduced operators use the spatial dimension as main condition. By this, the temporal dimension is adapted. Thus, spatio-temporal window operators have several advantages over conventional operators: They can reduce the number of elements in the result set, which lowers the memory and processing requirements. Furthermore, they support the formulation of spatial queries without the difficult task to estimate temporal conditions that often need an overestimation. Finally, they allow us to formulate queries with higher spatial semantics. The presented operators are especially useful for moving objects with changing speed.

In future work, we plan to evaluate the performance of the spatio-temporal operators. Furthermore, we have to consider the accuracy of the estimated vehicle location, which depends on multiple factors. Especially vehicles that cannot make use of the global navigation satellite system (GNSS) like AUVs are forced to locate themselves by alternatives. Submersible vehicles can make use of an acoustic based positioning system that is typically associated with higher inaccuracies than GNSS. In contrast to the introduced data set in Figure 2 locations can have a large deviation. As soon as a window is based on distances such inaccuracies can massively distort the calculated distances that may ends up in false results. Thus, we need an operator for preventing such situations.

The presented set of operators supplements existing temporal window types by introducing spatial capabilities. These operators are already sufficient for supporting several kinds of scenarios in the domain of moving vehicles. But, in some cases a combination of such operators would limit elements further to unburden the endpoint application. Therefore, efficient ways to build such cascades has to be developed.

Conventional spatial database systems are specialized on executing queries to select and provide data as fast as possible. Such queries consist of multiple operators that benefit massively from indexes that were built on table columns. Depending on the application, some of the presented windows can have a vast number of high dynamically elements. To speed up querying result sets based on spatial conditions there is a need for index structures that are specialized in high dynamic data. One approach is to adapt grid-based structures that are spatially fixed to avoid continuous index reorganisation processes.

References

1. Tholen, C., Nolle, L.: Parameter search for a small swarm of AUVs using particle swarm optimisation. Lecture Notes in Computer Science 10630 LNAI, 384–396 (2017)
2. Sliwka, J., Munafo, A., Petroccia, R.: Bandwidth Efficient Concurrent Localisation and Communication in Underwater Acoustic Networks. In: 2018 OCEANS - MTS/IEEE Kobe Techno-Oceans (OTO). pp. 1–7 (2018)
3. Carreras, M., Hernández, J.D., Vidal, E., Palomeras, N., Ridao, P.: Online motion planning for underwater inspection. In: 2016 IEEE/OES Autonomous Underwater Vehicles (AUV). pp. 336–341 (2016)
4. Paull, L., Saeedi, S., Seto, M., Li, H.: Sensor-Driven Online Coverage Planning for Autonomous Underwater Vehicles. IEEE/ASME Transactions on Mechatronics 18(6), 1827–1838 (2013)
5. Cudre-Mauroux, P., Wu, E., Madden, S.: TrajStore: An adaptive storage system for very large trajectory data sets. In: 2010 IEEE 26th International Conference on Data Engineering (ICDE 2010). IEEE (2010)
6. Wingerath, W., Ritter, N., Gessert, F.: Real-Time & Stream Data Management. Springer International Publishing (2019)
7. Carafoli, L., Mandreoli, F., Martoglia, R., Penzo, W.: Streaming Tables: Native Support to Streaming Data in DBMSs. IEEE Transactions on Systems, Man, and Cybernetics: Systems 47(10), 2768–2782 (2017)
8. Chandrasekaran, S., Cooper, O., Deshp, A., Franklin, M.J., Hellerstein, J.M., Hong, W., Krishnamurthy, S., Madden, S., Raman, V., Reiss, F., Shah, M.: TelegraphCQ: Continuous dataflow processing for an uncertain world. In: In First Biennial Conference on Innovative Data Systems Research (CIDR) (2003)
9. Gama, J., Rodrigues, P.: Data stream processing. Springer Berlin Heidelberg (2007)

10. Toth, C.K., Koppanyi, Z., Lenzano, M.G.: New Source of Geospatial Data: Crowd-sensing by Assisted and Autonomous Vehicle Technologies. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XLII-4/W8, 211–216 (2018)
11. Li, R., Ruan, S., Bao, J., Zheng, Y.: A Cloud-Based Trajectory Data Management System. In: Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - SIGSPATIAL17. ACM Press (2017)
12. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and Issues in Data Stream Systems. In: Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. pp. 1–16. PODS '02, ACM, New York, NY, USA (2002)
13. Akidau, T., Schmidt, E., Whittle, S., Bradshaw, R., Chambers, C., Chernyak, S., Fernández-Moctezuma, R.J., Lax, R., McVeety, S., Mills, D., Perry, F.: The dataflow model. Proceedings of the VLDB Endowment 8(12), 1792–1803 (2015)
14. Traub, J., Grulich, P.M., Cuellar, A.R., Bress, S., Katsifodimos, A., Rabl, T., Markl, V.: Scotty: Efficient Window Aggregation for Out-of-Order Stream Processing. In: 2018 IEEE 34th International Conference on Data Engineering (ICDE). IEEE (2018)
15. PipelineDB, Inc.: PipelineDB Documentation. http://docs.pipelinedb.com/ (2019), accessed: 2019-12-19
16. The Apache Software Foundation: Apache Flink 1.7 Documentation: Windows. https://ci.apache.org/projects/flink/flink-docs-release-1.7/dev/stream/operators/windows.html (2019), accessed: 2019-12-19
17. Appelrath, H.J., Geesen, D., Grawunder, M., Michelsen, T., Nicklas, D.: Odysseus: A Highly Customizable Framework for Creating Efficient Event Stream Management Systems. In: Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems. ACM Press (2012)
18. Chen, L., Shang, S., Yao, B., Zheng, K.: Spatio-temporal top-k term search over sliding window. World Wide Web (2018)
19. Chaccour, K., Assaad, H.A., el Hassani, A.H., Darazi, R., Andres, E.: Sway analysis and fall prediction method based on spatio-temporal sliding window technique. In: 2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom). IEEE (2016)
20. Nakamoto, Y., Okamoto, M., Bhuiya, M., Yamaguchi, A., Sato, K., Honda, S., Takada, H.: Android Platform Based on Vehicle Embedded Data Stream Processing. In: 2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing. pp. 48–55 (2013)
21. Werner, T., Brinkhoff, T.: Managing Spatio-Temporal Data Streams on AUVs. In: Proceedings of the IEEE/OES Autonomous Underwater Vehicles (AUV). IEEE, Porto, Portugal (2018)
22. Galić, Z.: Spatio-Temporal Continuous Queries. In: Spatio-Temporal Data Streams, pp. 17–45. Springer New York (2016)