Path overlap detection and property graph construction from TCX data for advanced analysis

1st Tilen Hliš
Faculty of Electrical Engineering and Computer Science, University of Maribor

2nd Iztok Fister
Faculty of Electrical Engineering and Computer Science, University of Maribor

3rd Iztok Fister Jr.
Faculty of Electrical Engineering and Computer Science, University of Maribor

2025 IEEE 23rd World Symposium on Applied Machine Intelligence and Informatics (SAMI) | 979-8-3503-7936-5/25/$31.00 ©2025 IEEE | DOI: 10.1109/SAMI63904.2025.10883235

***Abstract*—This paper presents a novel approach for processing cycling data from TCX files using a custom-built Julia library, `TCX2Graph.jl`. The method includes constructing property graphs from GPS track points and detecting overlapping path segments. These detected overlapping points are enriched with track point metrics, enabling detailed analysis of cycling sessions. The proposed approach processes large datasets efficiently, providing insights into overlapping paths and athlete performance.**

***Index Terms*—Path overlap, TCX files, property graph, GPS data, Julia.**

I. Introduction

Cycling, as both a recreational activity and a competitive sport, has been revolutionized by advancements in wearable technologies and Global Positioning System (GPS) devices. Athletes rely on data from training sessions (e.g., speed, distance, heart rate) to optimize performance. Training Center XML (TCX) files, a common format, provide insights into training effectiveness, track patterns, and areas for improvement.

Recent advancements, such as digital twins, have further improved athlete performance monitoring by creating virtual models that mirror real-world conditions [5]. Building on these innovations, this study focuses on analyzing cycling data with property graphs to detect overlapping paths and enrich these segments with performance metrics.

With the growing use of wearable fitness devices, vast amounts of data are generated daily, especially in formats such as TCX and GPS Exchange (GPX) format, which store detailed records of workouts, including GPS coordinates, heart rates, and other psychological metrics. These data hold significant potential for deeper insights, but their unstructured format and complexity pose challenges for advanced analysis.

The purpose of the paper is to represent the collection of the TCX files in the form of property graphs, that serve us for analysis of the performances of the athletes during the training sessions, which can happen on the same place. The property graphs consist of nodes and edges, including multiple attributes/properties. These are represented as directed multigraphs, whose nodes/edges are attached by the entities of the corresponding data structures. Typically, the properties are included in the form of a key-value. This representation allows us to detect path overlaps inside the TCX file simply.

Indeed, athletes producing the TCX files during the training sessions can overcome the same racetracks more times. In that case, the path overlaps are detected in the corresponding TCX files. This phenomenon has a crucial impact on detecting the behavior of the athlete in sports training. For instance, the athlete can overcome a specific part of the course, either at the beginning or the end of the training session. The physiological influence on the athlete's stress can be very different in both cases. In line with this, the characteristics of the athlete's behavior need to be indicated in the corresponding TCX files, i.e., path overlap detection and monitoring the performances of the athlete during the specific path. The athlete's behavior has a big influence on predicting the amount and intensity of the sports training session.

The contributions of this paper are as follow:

- to present a novel approach to reading, processing, and analyzing TCX files through a custom-built algorithm implemented in the Julia programming language called TCX2Graph.jl,
- to detect path overlaps inside the TCX files,
- to conduct two Case Studies, in order to verify a power of the proposed method,
- to use the mined knowledge to predict the athlete's behavior more accurately.

Obviously, the last paragraph is not the subject of the paper, but is left as an opinion for further work.

The rest of the paper is structured as follows: Section II covers the materials and methods, including description of TCX file formats, property graph construction, and path overlap detection. Section III details the proposed method and its implementation in the TCX2Graph.jl library. Section IV presents the experimental work, and corresponding performance analysis. Finally, Section V concludes with key contributions and outlines potential research directions.

II. Materials and methods

The purpose of this section is to present the data sources that were used in the developed algorithm, and the methods that were employed to extract insights from the data. We describe the data preprocessing steps, the construction of the property graph from the TCX data, and the algorithm used to detect path overlaps.

A. Format of the TCX files

The data used in the proposed algorithm consist of TCX files developed by Garmin for recording sports activities. The TCX files are employed by GPS-enabled fitness devices to store detailed workout data, such as time, distance, speed, heart rate, and cadence, in the XML format, making them flexible for various types of activity tracking [6, 3].

Each TCX file contains session data recorded at intervals (track points), including laps and GPS-stamped performance metrics, where key elements include:

- **Activity**: the overall session container, recording the sport type and metadata.
- **Laps**: segments of the session with data like time, distance, and heart rate.
- **Track points**: moment-specific records, including GPS coordinates, altitude, heart rate, cadence, and speed.

These TCX files, representing multiple cycling sessions, provide both performance statistics (speed, heart rate and others) and precise geographical information, allowing for advanced spatial analysis of overlapping paths. Each track point is georeferenced, enabling the comparison of routes and the detection of intersections between paths.

B. Property graphs

The property graph is defined as a 7-tuple:

$$\langle N, A, K, V, \alpha, \kappa, \pi \rangle, \tag{1}$$

where N is a set of nodes/vertices, A is a set of directed edges/arcs, K a set of keys in the form of attributes/properties, V a set of values attached to the keys, $\alpha : A \to N \times N$ is a label function defining the multi-graph property, κ a binary relation over $(A \cup N)$ and K, $\pi : \kappa \to V$ is a partial function providing values for the properties of the nodes and arcs which are including them.

III. The proposed method

Path overlap detection in property graphs is a critical challenge in analyzing the routes or paths followed by athletes. When multiple sessions of cycling are recorded and stored in TCX files, these paths can sometimes overlap. This overlap may occur when an athlete traverses the same route repeatedly in multiple sessions, or crosses an identical segment.

The problem can be described as identifying segments in the property graph where two or more paths share a common nodes (GPS track points), or are spatially close to each other within a given tolerance (accounting for GPS inaccuracies). The challenge is to query this graph efficiently to detect overlapping segments across different sessions or routes, while accounting for possible deviations in the recorded paths due to GPS drift.

The main goal of path overlap detection is to recognize accurately when different rides follow the same or similar physical routes, and then analyze the performance metrics (such as speed, heart rate, and cadence) across these shared segments. This helps provide insights into an athlete's consistency, identify performance patterns, and account for external factors like wind, terrain, or fatigue, that might affect performance in overlapping segments.

The proposed method consists of three steps as follows:

- parsing of the TCX files,
- constructing the property graph,
- path overlap detection.

The mentioned steps are presented in detail in the remainder of the paper.

A. The effective parsing of the TCX files

The first step in our developed algorithm, called `TCX2Graph.jl`, is the efficient reading and parsing of the TCX files. We implemented a custom solution to process these files using the `TCXReader.jl` library, which we developed in the Julia programming language [4].

The TCX data extracted by `TCXReader.jl` is organized into three primary structures: `TCXActivity`, `TCXLap`, and `TCXTrackPoint`. These structures facilitate detailed parsing and analysis of the workout data, as follows:

- **TCXActivity**: The top-level structure represents the entire workout session, containing metadata such as sport type and a unique session ID. It stores multiple `TCXLap` structures with aggregated metrics like total time and distance.
- **TCXLap**: Divides each activity into laps, storing data like start time, distance, speed, and heart rate. Each lap contains multiple `TCXTrackPoint` records for tracking performance over time.
- **TCXTrackPoint**: The most granular structure, representing specific moments in the workout, capturing performance metrics, and GPS coordinates (Listing 1).

Listing 1. The `TCXTrackPoint` struct from `TCXReader.jl`

```
struct TCXTrackPoint
    time::DateTime
    latitude::Union{Nothing, Float64}
    longitude::Union{Nothing, Float64}
    altitude_meters::Union{Nothing, Float64}
    distance_meters::Union{Nothing, Float64}
    heart_rate_bpm::Union{Nothing, Int}
    cadence::Union{Nothing, Int}
    speed::Union{Nothing, Float64}
    watts::Union{Nothing, Int}
end
```

In addition to these primary structures, `TCXReader.jl` employs `TCXAuthor` and `DeviceInfo` to capture metadata related to the device and author of the TCX file. However, the core analysis focuses on `TCXActivity`, `TCXLap`, and `TCXTrackPoint`, which, together, form the backbone of the TCX data extraction process, enabling further analysis of overlapping segments in the cycling routes.

B. Constructing the property graph

After reading and parsing the TCX files, the next step in our approach is the construction of a property graph to represent the cycling data from multiple training sessions. This property graph, generated using the `Graphs` library [7] in

Julia within our `TCX2Graph.jl`, organizes the GPS points from the cycling sessions into vertices, with edges connecting consecutive points along each session's path. The property graph serves as an effective means to store and manage the various track point-related metrics. The property graph consists of three primary components:

- **A simple graph**: This graph represents each GPS point as a vertex, and the edges connect consecutive points from the same session, preserving the path structure.
- **A dictionary of GPS data**: This dictionary stores detailed properties for each vertex, such as latitude, longitude, and other track point-specific metrics extracted from the TCX files.
- **A vector of paths**: This vector holds the ranges of vertex indices corresponding to each individual session, allowing for easy tracking of the paths across multiple TCX files.

In constructing this property graph, the GPS points are processed by first adding each point as a vertex, followed by connecting consecutive points with edges, thus forming the paths. Simultaneously, the GPS properties are stored in the dictionary for fast access during further analysis.

While the property graph is crucial for organizing and accessing the GPS and track point data, it is not used directly for the identification of overlapping segments between paths, as performing spatial searches on a graph structure would be computationally expensive. Instead, we utilized a `KDTree` library [8] for efficient spatial querying, to detect overlapping segments across different paths. Once overlapping segments are identified, the property graph is then employed to retrieve the associated metrics from the GPS points within those segments.

Figure 1 shows a graphical visualization of the property graph constructed from the TCX data, with each path representing a cycling session from the input files.

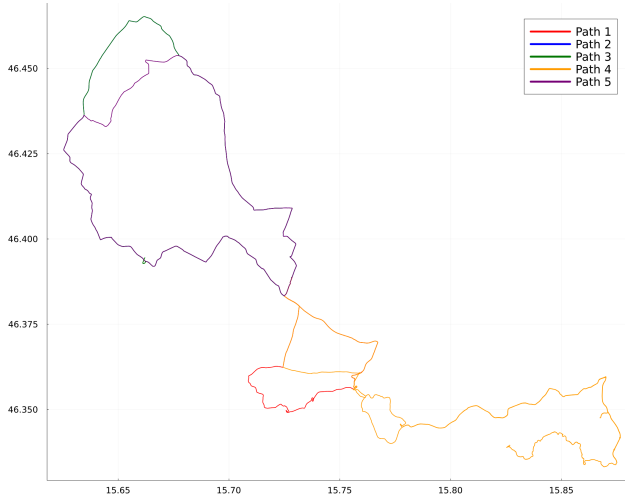

Fig. 1. Graphical visualization of property graph constructed from the TCX data.

In the next section, we detail the process of detecting overlapping segments using a `KDTree`, and how the property graph is used to enrich the overlapping segments with additional metrics.

C. Path overlap detection

After constructing the property graph, the next step in our algorithm is the detection of overlapping segments across different cycling paths. Given the inherent inaccuracies in GPS data, cyclists following the same physical route may have slight variations in the recorded locations. These variations necessitate a method that accounts for this inaccuracy when identifying overlapping segments. Our approach leverages a `KDTree` [8] structure to manage spatial queries efficiently, and implement a tolerance-based method to detect overlaps.

A KDTree, which is a data structure optimized for nearest-neighbor queries, allows for efficient spatial searches within our GPS dataset. This data structure has proven effective for high-dimensional data in real-time applications [10, 1, 9]. We chose the KDTree library in Julia due to its efficiency and active maintenance [8]. By organizing the GPS points into this tree, we can identify points that are spatially close to each other across different paths quickly. The primary function, `find_overlapping_segments_across_paths` (see Algorithm 1), identifies overlapping segments by comparing each point in a path to nearby points in other paths. A tolerance radius is applied, to accommodate the slight deviations that occur due to GPS inaccuracy. This tolerance ensures that paths that follow the same route but have minor spatial offsets are still recognized as overlapping.

For each pair of paths, the algorithm checks whether points from different paths fall within each other's tolerance radius. If points fall within this radius for a sufficient consecutive number of points, they are considered part of an overlapping segment. The algorithm updates the mapping of overlapping segments, ensuring that only significant overlaps are recorded.

Figure 2 provides a graphical representation of how tolerance is applied when detecting overlapping paths, highlighting the flexibility in identifying common routes even in the presence of minor GPS discrepancies.

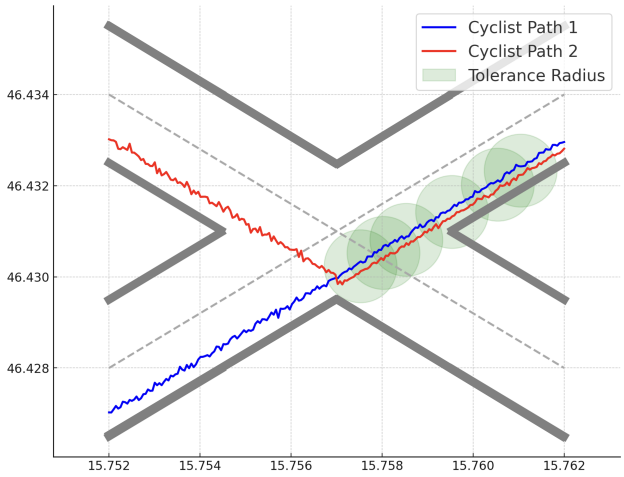

Fig. 2. Identification of overlapping segment with tolerance.

Algorithm 1 find_overlapping_segments_across_paths

Require: all_gps_data, paths, kdtree, max_gap, min_segment_length, segment_gap_tolerance
Ensure: Return a list of overlapping segments

```
Initialize overlap_segments and segment_map
for all path_idx in paths do
    Initialize segment_start, current_segment, gap_count, overlapping_paths
    for all idx in path do
        gps1 ← gps_to_point(all_gps_data[idx])
        candidates ← inrange(kdtree, gps1, max_gap)
        found_overlap ← False
        for all candidate_idx in candidates do
            if candidate_idx is not in path then
                Continue
            end if
            for all other_path in paths do
                if candidate_idx in other_path and is_same_location(...) then
                    found_overlap ← True, gap_count ← 0
                    Append idx to current_segment, update segment_map
                end if
            end for
        end for
        if not found_overlap then
            Increment gap_count
            if gap_count > segment_gap_tolerance and valid segment then
                Append segment to overlap_segments, reset variables
            end if
        end if
    end for
    if valid segment at end of path then
        Append segment to overlap_segments
    end if
end for
return overlap_segments
```

This approach, combined with the KDTree, ensures that the path overlap detection process is computationally efficient, even when processing large datasets, such as, for instance, the 462 TCX files as used in our tests.

Once the overlapping segments are identified, we use the previously constructed property graph to extract and enrich the overlapping segments with additional track point metrics such as speed, heart rate, altitude, and cadence (see Listing 1 for a full list of metrics). This data extraction step is crucial for enabling a more in-depth analysis of cyclist performance across these shared segments. These enriched overlapping segments are then prepared for further analysis, such as generating transactions for use in Numerical Association Rule Mining, which focuses on mining numerical relationships between track point metrics. The Numerical Association Rule Mining methodology represents a future goal in our research, which is to perform advanced pattern discovery in the cycling data.

IV. Experiments and results

The goal of this experimental work is to evaluate the performance of the proposed path overlap detection algorithm in identifying overlapping cycling segments from TCX files. Specifically, we aim to assess the algorithm's efficiency in processing multiple TCX files, detecting path overlaps, and providing insightful performance metrics for athletes. The focus is on using property graphs and KDTree spatial queries to extract overlapping paths efficiently, and enrich them with cycling performance data.

The algorithm, implemented in Julia using `TCX2Graph.jl`, was tested on a MacBook Pro M3 (18GB RAM)(Table I) with 462 TCX files [2]. Of these, five files recorded by the same cyclist were selected for analysis, revealing 39 overlapping segments, with two segments chosen for detailed study.

TABLE I
Performance metrics for overlap detection.

Metric	Value
Threads	5
Runtime (s)	1.03
Allocations (M)	27.52
Memory (MiB)	870.37

For further analysis and experiment, we selected two segments manually:

- **Segment 30**, which has been traversed across five separate rides.
- **Segment 18**, which has been traversed across three separate rides.

The manual selection of files enabled verification of conditions during the rides, such as stronger winds reported on Strava, which explained slower performance on overlapping paths. The algorithm accurately identified all overlapping segments in the selected files, with no false positives or negatives. Key statistics (e.g., minimum, maximum, mean, median, and Standard deviation (Stdev)) for metrics like speed, Heart Rate (HR), cadence, and time provided insights into the cyclist's performance consistency and physiological response.

A. Results

In order to show the correctness of our Research Questions set at the beginning of the experimental work, two Case Studies were conducted as follows:

- Case Study 1: Path overlap detection from a property graph consisting of five sport training activities.
- Case Study 2: Path overlap detection from a property graph consisting of three sport training activities.

In the remainder of the paper, both Case Studies are illustrated in detail.

1) Case study 1: Research Question 1 (RQ-1) in Case study 1 was set as: " The cyclist's performance on overlapping segments is impacted significantly by external factors such as wind, resulting in a higher variance in speed and heart rate across multiple rides.". In line with this, the cyclist rode the path denoted as Segment 30 five times across different sessions. Table II presents the aggregated statistics for each metric.

TABLE II
PERFORMANCE METRICS FOR SEGMENT 30 AFTER FIVE RIDES.

Metric	Min	Max	Mean	Median	Stdev
Speed [km/h]	22.5	25.2	24.16	24.8	1.19
HR [bpm]	143	165	151.6	147	9.48
Cadence [rpm]	79	86	83.4	85	2.88
Time [sec]	1,539	1,724	1,607.6	1,563	81.43

Figure 3 shows the visualization of the overlapping Segment 30. Thus, Table III provides the observed characteristics

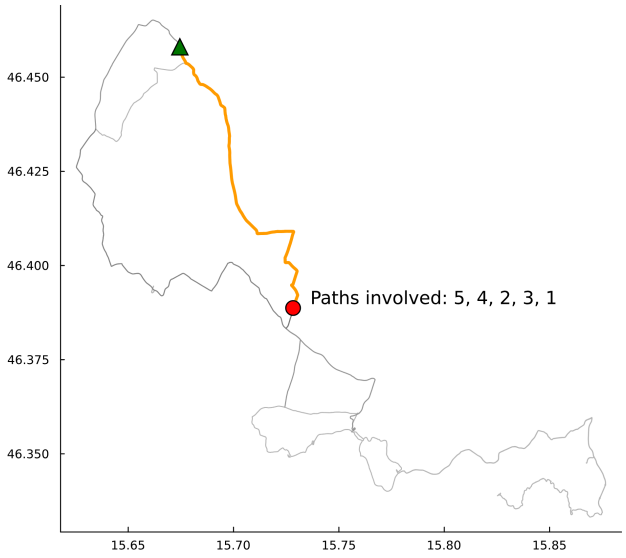

Fig. 3. Graphical visualization of overlapping Segment 30 in the cycling routes.

of Segment 30, including distance, ascent, descent, maximum gradient, and average gradient.

An analysis of the results for Segment 30 revealed a consistent cycling performance with a relatively small standard deviation in speed (**Stdev**=1.19 km/h), suggesting that the cyclist maintained a steady pace across the five rides. The heart rate variability was more pronounced, with a range $\mathbf{HR} \in [143, 165]$ bpm, possibly reflecting the varying levels of exertion or fatigue. The variance in heart rate (i.e., HR $= 89.8$ bpm) suggests that external factors, such as wind, influenced the cyclist's performance, which we confirmed via

TABLE III
CHARACTERISTICS OF THE OBSERVED SEGMENT 30.

Characteristic	Value
Segment ID	30
Distance	10,768.57 meters
Ascent	34.00 meters
Descent	57.00 meters
Max Gradient	20.19 %
Average Gradient	2.99 %

the cyclist's Strava data. Specifically, the rides that showed slower speeds corresponded to reports of stronger winds.

The standard deviation in time $\mathbf{Stdev} = 81.43$ sec also indicates variability in how long it took the cyclist to complete this segment, with a range of approximately three minutes between the fastest and slowest rides.

B. Case study 2

Research Question 2 (RQ-2) in Case study 2 was set as: "The cyclist's performance on overlapping segments is less affected by external factors, resulting in lower variance in key performance metrics Speed, HR, and Cadence across rides.". In line with this, the cyclist rode the path denoted as Segment 18 three times across different sessions. Table IV presents the aggregated statistics for each metric, while Figure 4 shows the visualization of overlapping segment 18, Table V provides the observed characteristics of Segment 18, including distance, ascent, descent, max gradient, and average gradient.

TABLE IV
PERFORMANCE METRICS FOR SEGMENT 18 AFTER THREE RIDES.

Metric	Min	Max	Mean	Median	Stdev
Speed [km/h]	21.5	22.3	21.93	22	0.40
HR []bpm]	162	166	163.67	163	2.08
Cadence [rpm]	75	81	78.33	79	3.06
Time [sec]	910	945	926.67	925	17.56

As is evident from the results in Table V, the segment 18 showed a more stable performance across the rides. The cyclist's speed and cadence in Table IV were highly consistent, with low standard deviation in the metrics Speed and Cadence (i.e., $\mathbf{Stdev} = 0.40$ km/h for Speed and $\mathbf{Stdev} = 3.06$ rpm for Cadence). This suggests that the cyclist was able to maintain a similar level of effort across all the rides. The heart rate shows slightly more variability, but the range HR $\in [162, 166]$ bpm was small indicating a relatively uniform physiological response. The standard deviation in time

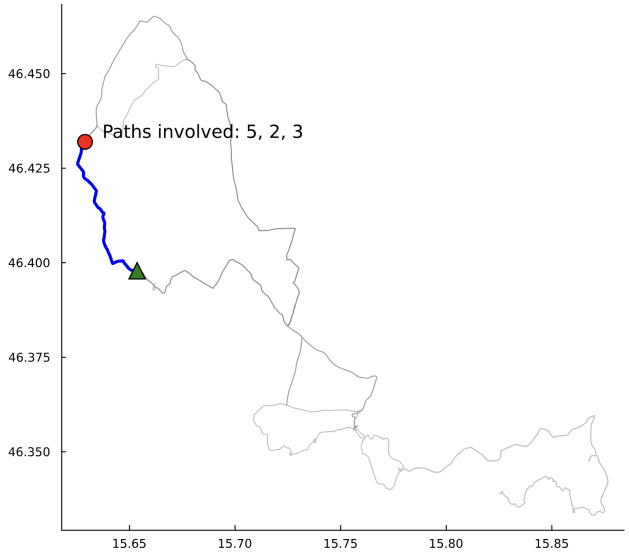

Fig. 4. Graphical visualization of overlapping Segment 18 in the cycling routes.

TABLE V
CHARACTERISTICS OF THE OBSERVED SEGMENT 18.

Characteristic	**Value**
Segment ID	18
Distance	5,243.09 meters
Ascent	50.20 meters
Descent	16.60 meters
Max Gradient	18.55%
Average Gradient	2.91%

was also low (i.e., $\mathbf{Stdev} = 17.56$ sec) suggesting that the cyclist's performance on Segment 18 was more predictable and consistent compared to Segment 30.

In summary, the results support both set research questions (i.e., RQ-1 and RQ-2), as Segment 30 exhibited higher variance in both speed and heart rate, likely due to varying wind conditions, while Segment 18 demonstrated more consistent performance.

V. CONCLUSION

This paper introduced a novel approach for processing cycling data from TCX files using the Julia-based `TCX2Graph.jl` library. We demonstrated how to construct property graphs from GPS track points and detect overlapping segments in cycling routes using a `KDTree` for spatial querying.

In our experimental study, we selected two overlapping segments manually from a subset of five TCX files, all provided by the same cyclist. Detailed performance metrics, including Speed, HR, Cadence, and Time, were calculated for each segment. The results showed significant variability in Segment 30, where external factors such as wind impacted the cyclist's performance, as evidenced by the variance in speed and heart rate. In contrast, Segment 18 exhibited more consistent performance across all the metrics, confirming our research questions that external factors had less of an influence on this segment. Analyzing overlapping segments and variance provided valuable insights into cyclist behavior, highlighting the impact of external conditions on performance.

Future work would explore the integration of the extracted performance data into advanced transaction modeling, such as Numerical Association Rule Mining, to discover significant patterns and rules in cycling performance. To improve the generalizability and robustness of the results, future research would expand the dataset to include information from multiple cyclists, diverse geographical regions, and varied cycling conditions. In general, the more predictive modeling that considers also external factors, like terrain and weather, could results in a universal framework for evaluating and forecasting sport performance using TCX files.

ACKNOWLEDGMENT

The authors acknowledge the financial support from the Slovenian Research Agency (Research Core Funding No. P2-0057).

REFERENCES

[1] Yewang Chen et al. "Fast neighbor search by using revised kd tree". In: *Information Sciences* 472 (2019), pp. 145–162.

[2] Iztok Fister et al. *A collection of sport activity datasets with an emphasis on powermeter data*. University of Ljubljana, 2017.

[3] *Garmin developer program*. https://developer.garmin.com/gc-developer-program. [Accessed 06-09-2024].

[4] Tilen Hliš and Iztok Fister. *TCXReader.jl*. https://juliahub.com/ui/Packages/General/TCXReader. [Accessed 06-09-2024]. 2024.

[5] Tilen Hliš, Iztok Fister, and Iztok Fister Jr. "Digital twins in sport: Concepts, taxonomies, challenges and practical potentials". In: *Expert Systems with Applications* 258 (2024), p. 125104.

[6] Kashif Iqbal. *TCX File Format- Training Center XML File — docs.fileformat.com*. https://docs.fileformat.com/gis/tcx/. [Accessed 06-09-2024].

[7] JuliaGraphs. *Graphs*. https://juliahub.com/ui/Packages/General/Graphs. [Accessed 07-10-2024].

[8] KristofferC. *NearestNeighbors*. https://juliahub.com/ui/Packages/General/NearestNeighbors. [Accessed 07-10-2024].

[9] Parikshit Ram and Kaushik Sinha. "Revisiting kd-tree for nearest neighbor search". In: *Proceedings of the 25th acm sigkdd international conference on knowledge discovery & data mining*. 2019, pp. 1378–1388.

[10] Kun Zhou et al. "Real-time kd-tree construction on graphics hardware". In: *ACM Transactions on Graphics (TOG)* 27.5 (2008), pp. 1–11.