A Graph-based Approach to Human Activity Recognition

Thomas Peroutka*, Ilir Murturi*, Praveen Kumar Donta†, and Schahram Dustdar*
*Distributed Systems Group, TU Wien, Vienna 1040, Austria.
†Department of Computer and Systems Sciences, Stockholm University, Stockholm 16425, Sweden.

***Abstract*—Advanced wearable sensor devices have enabled the recording of vast amounts of movement data from individuals regarding their physical activities. This data offers valuable insights that enhance our understanding of how physical activities contribute to improved physical health and overall quality of life. Consequently, there is a growing need for efficient methods to extract significant insights from these rapidly expanding real-time datasets. This paper presents a methodology to efficiently extract substantial insights from these expanding datasets, focusing on professional sports but applicable to various human activities. By utilizing data from Inertial Measurement Units (IMU) and Global Navigation Satellite Systems (GNSS) receivers, athletic performance can be analyzed using directed graphs to encode knowledge of complex movements. Our approach is demonstrated on biathlon data and detects specific points of interest and complex movement sequences, facilitating the comparison and analysis of human physical performance.**

***Index Terms*—Human Activity Recognition, IoT, GNSS, Graphs**

I. Introduction

Throughout human evolution, our bodies and brains learned the ever-increasing motion complexity. Even decades into the computer and machine-powered area, it is still hard for machines to solve supposedly easy tasks like manipulating a Rubik's cube [1]. This means humans and all living things have the unique ability to move their bodies accurately and efficiently. With more and more wearable sensors available, such movements can be measured precisely and gain insights into efficiency. The wearable technology market is predicted to grow at a compound annual growth rate of 14.6% from 2023 to 2030 [2]. This includes using standard hardware like "Cardiovascular Monitoring Using Earphones and a Mobile Device" [3] or "A Wearable Sensor for Measuring Sweat Rate" [4]. There are even sensors that act as little radar to detect human activities based on millimeter-waves [5]. All these sensors generate a large amount of data with a common objective: detecting physical activities and extracting insights. This field of study is known as Human Activity/Action Recognition (HAR) [6].

The HAR field researches how to identify and understand human activities using technology. Despite the progress made in this field, crucial aspects should be addressed to significantly transform how individuals interact with mobile devices and other devices. In the last few years, there have been extensive studies in multiple areas, such as 3D Convolutional Neural Networks (CNN) for HAR. Ji et al. [7] demonstrated how image-based activity recognition is possible. In contrast, Bia et al. [8] showcases how a wearable sensor and a CNN can detect human activity like walking, sitting, or climbing stairs. Another approach is a statistical analysis of given data and trying to classify the activities [9]. The current research aims to interpret datasets without prior knowledge, meaning that they do not describe what movements look like to the system. Meanwhile, most of these approaches are resource-intensive (using neural networks) or are difficult to set up. Therefore, our goal in this paper is to use domain-specific knowledge (i.e., anatomy of a movement sequence) obtained by experts (e.g., trainers) and encoded into a directed graph to quickly and reliably detect motion sequences (i.e., movements). Furthermore, we emphasize using existing algorithms, which make analyzing big datasets efficient and resource-friendly. Nevertheless, our approach is limited to human activity recognition; however, it can also be used in any time data series.

In this paper, we propose an approach that aims to represent complex athletes' movements as a directed graph. The goal is to find an efficient method for identifying critical points in a multi-sensor dataset and detecting complex movements. This approach is versatile and can be applied to any movement. For illustration, we will focus on biathlon, a demanding sport that combines two very different activities: fast skiing on various terrains and precise target shooting while stationary. Speed is crucial in biathlon, so finding the most effective and efficient body movement techniques is essential. This involves determining the best approach for uphill, flat, or downhill sections and maximizing shooting accuracy. Moreover, we aim to identify a data structure capable of capturing body movements and developing an algorithm that can efficiently detect these movements using multiple wearable sensors. Ultimately, we will obtain performance indicators specific to different movements that can be compared with others. This will enable professional athletes, sports enthusiasts, and rehabilitation patients to enhance their physical movements. Note that this work focuses on the technological challenges associated with detecting user-defined motion sequences, and any sports science insights provided in this article are purely for illustrative purposes and may not be scientifically grounded.

The remaining sections are structured as follows. A motivation example and related work are presented in Section II. Section III presents the proposed approach and an example

to illustrate how to represent a sequence of movements in a directed graph. Evaluation results are discussed in Section IV. Lastly, we conclude our discussion with possible future actions in Section V.

II. Motivation and Related Work

A. Motivation Scenario

Understanding and improving athlete performance is critical in professional sports (i.e., especially in disciplines like biathlon). Biathlon combines cross-country skiing and rifle shooting, demanding peak physical condition and precision. Coaches and sports researchers constantly seek new methods to analyze and optimize the performance of athletes. Imagine a biathlon team preparing for the winter sports. The team's performance analytics division provides detailed insights into each athlete's performance to identify strengths and areas for improvement.

Analyzing transitions between race stages helps to identify athlete delays and fatigue signs (e.g., observing how athletes manage the shift from starting to tackling the uphill challenge, and then from the uphill to the shooting range). Traditional performance analysis methods are time-consuming and often lack the granularity to make such fine-tuned adjustments. Using IMUs and GNSS receivers, the responsible team can collect detailed movement data from athletes during training and competitions. Directed graphs allow encoding complex movement sequences, which in return allows the team to (i) capture detailed movement patterns, (ii) identify possible performance bottlenecks, (iii) optimize training, or (iv) compare athlete performances.

B. Related Work

HAR is a rapidly evolving field with various approaches to tracking and interpreting human actions [10]. A common methodology involves analyzing images and videos to detect and extract human figures and their movements [11]–[13]. The authors present a real-time system for 3D human pose estimation, tracking, and recognition from RGB-D video sequences using a generative structured framework. Kaya et al. [14] proposed a new 1D-CNN-based deep learning approach for sensor-based HAR using raw accelerometer and gyroscope data. The proposed work showed the impact of using individual sensor data versus combined data while finding that the model performed better with concurrent sensor data. Liu et al. [15] applied the Bayesian structural time series framework to biomedical sensor data, demonstrating its ability to accurately assess the significance of health interventions while accounting for complex covariate structures. The developed tool called MhealthCI processes and registers diverse biomedical data for personalized medicine applications. Kumar et al. [16] propose a novel Deep-HAR model that combines CNNs for feature extraction and Recurrent Neural Networks (RNNs) for pattern recognition in time-series data. On the contrary to these works, the innovative technique utilizes WiFi technology to ascertain human presence without the need for cameras or sensors [17].

Contrary to the above-mentioned works, our approach focuses on the most common way to use inertial sensors in wearable devices. The majority of research is concerned with classification, but very little is done in extracting more than just classes from datasets. Therefore, we aim to take it a step further and not only classify activities (e.g., smartwatches or smartphones classify human activities [18]) but also be able to extract user-defined performance metrics like forces applied to the body during certain tasks, etc. Our proposed approach requires domain-specific knowledge, requiring an understanding of how to break down a complex movement into key points. In contrast to other research works, our proposed graph-based method combined with domain-specific knowledge enables complex movement detection in a resource-efficient manner. Furthermore, our approach generates explainable results (meaning it is possible to reason (i.e., results are traceable and how they were calculated), which most approaches, especially ones using neural networks, lack. The advantages of our approach arise from the combination of a strict rule set driven by domain-specific knowledge and the expressive capabilities of graphs. This unique combination provides key benefits to our approach.

III. The proposed approach

A. An overview of the approach

In Figure 1, we outline the concept of the proposed approach. The first step is to break down a specific movement into smaller parts that make up the movement. This is done with the help of an expert (i.e., typically trainers or sports scientists). This step, e.g., includes trainers setting up virtual gates on a map or defining triggers for acceleration data. The temporal dependency between those parts is encoded into a directed graph. Movements are recorded with wearable devices featuring many sensors (e.g., IMU, GNSS, heart rate sensor, etc.). The data is then processed by our proposed approach (i.e., described in the next sections) and trainers receive insights about the movements of interest.

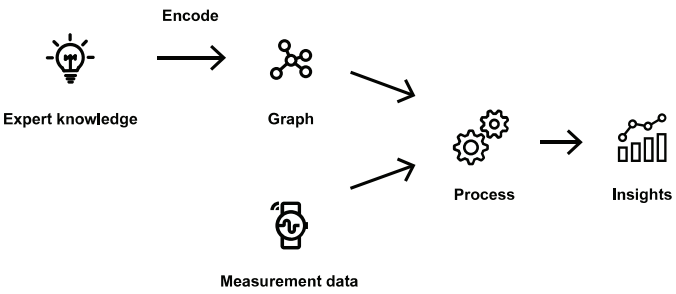

Fig. 1: An overview of the proposed approach.

B. Encoding complex movements into a directed graph

Biathlon and any other movements can be modeled as sections of movement that can be divided into smaller and smaller subsections. The macro-view (see Figure 2) tells us the shooting sequence consists of entering the shooting range

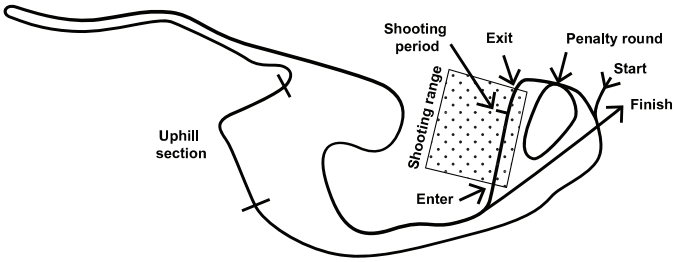

Fig. 2: A typical biathlon racetrack layout.

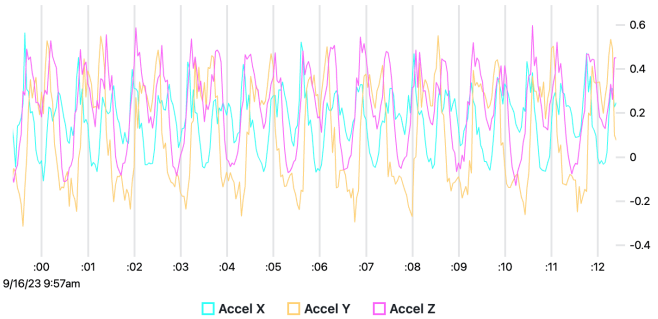

Fig. 3: Acceleration data from an uphill section.

followed by a shooting period and ending by exiting the shooting range and starting running the uphill section. If the uphill section is divided into multiple subsections, the analysis can be done at a micro level (refer to Figure 3). It shows the precise sequence of motion to complete one uphill step. First, the arms of the athletes are moved forward, then the upper body leans forward, and the feet follow. This sequence always happens in the same order but with different forces and duration. Such insight gained represents the knowledge of this movement and should be encoded in a directed graph.

After a brief analysis of motion in biathlon, the sequence of movements and their temporal dependency emerge as perfect properties for describing motion sequences. The simplest movement consists of two actions (i.e., referred to as points of interest) and can be noted as $A \rightarrow B$, which means A is a condition for B (or simply said, A needs to happen before B). Taking the example from above, one can write $UE \rightarrow UL$ where UE = uphill enter and UL = uphill leave/exit, meaning that exiting the uphill section could only happen after the uphill section was entered in the first place. This way, all temporal dependencies are simply defined as:

Definitions:

S	Start
UE	Enter uphill
UL	Exit/leave uphill
P	Penalty round
F	Finish
RE	Enter shooting range
SS	Start shooting
SF	Finish shooting
RL	Leave/exit shooting range

Dependencies:

$S \rightarrow UE$	(Start to entering uphill)
$UE \rightarrow UL$	(Entering uphill to exiting uphill)
$UL \rightarrow RE$	(Exiting uphill to entering shooting range)
$UL \rightarrow F$	(Exiting uphill to finish line)
$RE \rightarrow RL$	(Entering shooting range to leaving shooting r.)
$RE \rightarrow SS$	(Entering shooting range to start shooting)
$SS \rightarrow SF$	(Start shooting to finish shooting)
$SF \rightarrow RL$	(Finish shooting to leaving shooting range)
$RL \rightarrow P$	(Leaving shooting range to penalty round)
$P \rightarrow P$	(Penalty round to another penalty round)
$P \rightarrow UE$	(Penalty round to entering uphill)

Those dependencies can be placed into one of the most common and well-studied data structures in computer science known as graphs [19]. In this case, directed graphs will be utilized to encode the temporal dependencies. A directed graph is defined as an ordered pair $G = (V, E)$ where

- V is a set whose elements are called vertices or nodes. The representations will be used to depict actions or points of interest, such as reaching a specific speed or experiencing acceleration in a particular direction.
- E is a set of ordered pairs of vertices called directed edges. The representations will be used to illustrate the temporal dependencies of the points of interest mentioned in the example with macro and micro-views in biathlon.

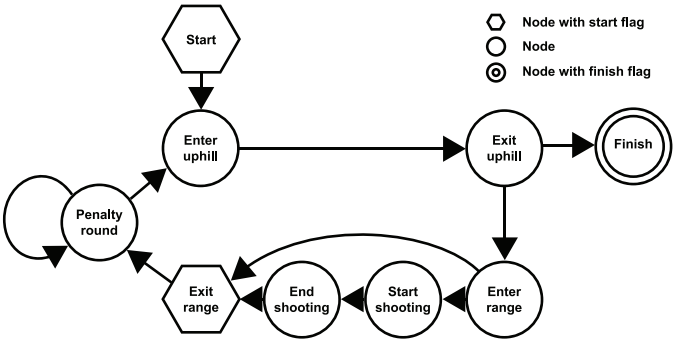

Fig. 4: Macro-view of biathlon. Turning sequence of events into a directed graph.

Nodes represent specific points of interest, like entering or exiting an area or the start of a specific movement. The directed edges encode the order depending on time. In this manner, any sequence and, thus, any kind of human activity can be encoded into a data structure well understood by computers.

C. Detection of points of interest in multi-sensor datasets

With datasets consisting of millions of data points, it is very inefficient to analyze each data point. The goal is to identify

specific points in time that signal the occurrence of events required to detect a more complex motion. Given that most wearables record data from various sensors, it is important to develop methods for detecting both generic events and sensor-specific events. The most common sensor consists of an inertial measurement unit (IMU) sensor (which delivers acceleration and angular velocity data) and a global navigation satellite system (GNSS) receiver (which delivers position and speed data). The following triggers are helpful to detect points of interest in big data sets:

1) Generic triggers:

- Edge detection: Falling, rising, or change (both rising and falling) edge commonly used in signal processing. Used in devices like oscilloscopes to detect events. The

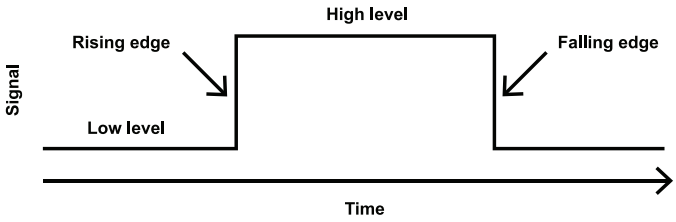

Fig. 5: Edge detection anatomy.

simplest implementation looks at a specific threshold. Mathematically, this function can be described as:

$$f(x) = \begin{cases} 1 & x \geq threshold \\ 0 & otherwise \end{cases}$$

- Peak detection: Detects peaks by comparing local minima/maxima with neighboring values. There are many

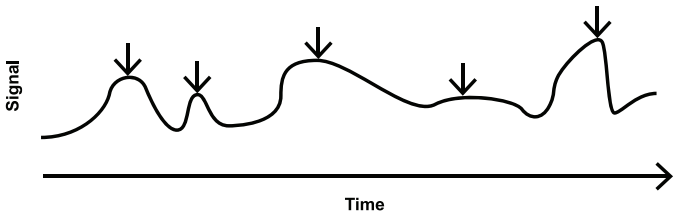

Fig. 6: Peak detection anatomy.

different algorithms to detect peaks. Some relevant examples are focused on electrocardiography [20], and there are even specialized ones [21] on detecting heartbeats. Nevertheless, we will not go into details about their inner workings as this is out of the scope of this work. Note that generic triggers can be used on any numerical dataset.

2) Sensor-specific triggers:

- Location-based detection: Detects when a line (virtual gate) is crossed (intersection), or areas are entered/exited. Note that sensor-specific triggers can only be used on specific datasets.
  There are multiple ways to calculate intersections. It depends on the given coordinates. In a Cartesian coordinate system, the two line equations $y = ax + c$ and $y = bx + d$ are checked with simple rearranging and substitutions if and where those lines intersect. The complexity increases when considering the geographic coordinate system used in GNSS coordinates. Hence, it is essential to consider the model used to represent Earth. The Earth is commonly depicted as an ellipsoid, often with the WGS84 model, which is also used by the Global Positioning System (GPS) [22]. For example, a simple formula to calculate the distance on a sphere is the Haversine formula [23]. One can also project the geographic coordinates to a Cartesian coordinate system and do calculations as mentioned above, but this only works for small distances.

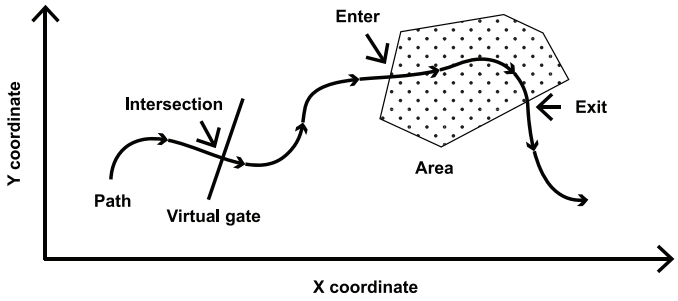

Fig. 7: Location-based detection anatomy.

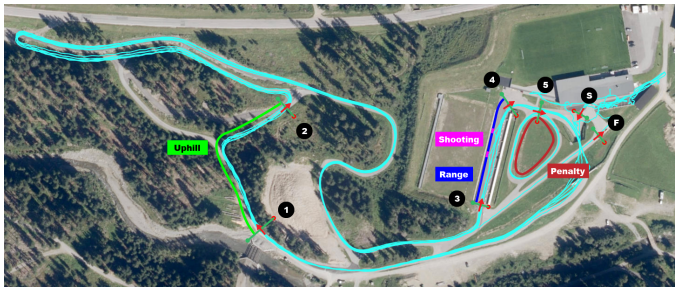

Fig. 8: Virtual gates track layout.

To measure a typical biathlon race, location-based detections, often referred to as virtual gates, are utilized. To better illustrate how the proposed algorithm works, the following sample data set will be used. The virtual gate S in Figure 8 marks the start line. Next will be the uphill section consisting of gates 1 and 2. Then, athletes will enter the range enclosed by gates 3 and 4. The shooting section in the range will be detected based on the measurement of the speed when lying down (lower than 1m/s) and getting back up again (higher than 1m/s). Then, for each missed shot out of 5 total shots, athletes need to run the penalty round. So, if an athlete misses 2 out of 5, he must do two penalty rounds. Gate number 4 is special in that it marks the beginning/ending of a lap. In total, six laps were done. Shooting is only done every 2nd lap.

A point of interest (POI) refers to a specific moment in time (timestamp) of an event that is being measured. For example, gate 1 will produce a POI each time the athlete crosses the

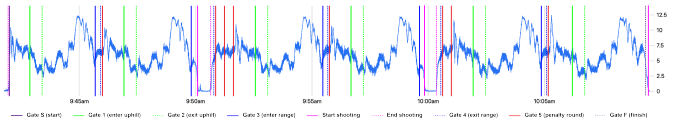

Fig. 9: Triggers plotted in chart with speed data (m/s).

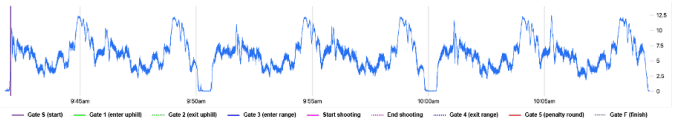

Fig. 10: All points of interest from start nodes – in our example, only one.

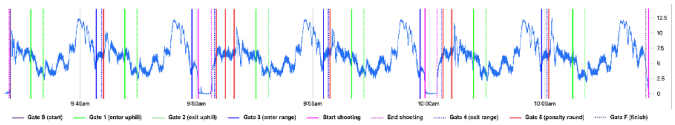

Fig. 11: POIs from different nodes used as input alphabet for our DFA.

virtual line. The following notation is used for a POI: S_t = point of interest S (start) was triggered at timestamp t.

Figure 9 shows all POIs (vertical lines) plotted with athlete speed in m/s. All POIs are triggered by virtual lines except start/end shooting. The speed information triggers those. If athletes move more than 1m/s, it will trigger on the falling edge (start shooting) or rising edge (end shooting). A closer look at the first and last POI shows that those are "wrongly" start and end shooting triggers because triggers are unaware of any temporal dependency (as previously described).

D. A direct graph to recognize complex movements

Up to now, the complex movement of interest is encoded into a directed graph and points of interest from our dataset. It is a matter of applying the graph to the points of interest to recognize our complex movement. In order to accomplish this, the graph needs to be traversed based on the points of interest. This problem can be translated into a Deterministic Finite Automaton (DFA) to explain this step more clearly. A DFA is described by a five-element tuple $(Q, \sum, \delta, q0, F)$ [24]:

Q	states
$\sum$	input alphabet
δ	transition functions
$q0$	the starting state
F	accepting states

The standard definition of $q0$ needs to be modified to allow for multiple starting states, considering the possibility of having multiple start nodes. A start node is defined as a node in our directed graph (as shown in Figure 4) marked with a "start" flag. The following DFA is deduced from our example of biathlon:

$$DFA_{Biathlon}(Q, \sum, \delta, q0, F)$$

Firstly, $Q = \{S, UE, UL, P, RE, RL, SS, SF, F\}$ represents all nodes of the directed graph. Secondly, $\sum = \{SF_{163}, S_{175}, UE_{214}, UL_{235}, \dots\}$ represent all points of interest from Section III-B. Lastly, $\delta = \{S \rightarrow UE, UE \rightarrow UL, UL \rightarrow RE, UL \rightarrow F, RE \rightarrow RL, RE \rightarrow SS, SS \rightarrow SF, SF \rightarrow RL, RL \rightarrow P, P \rightarrow P, P \rightarrow UE\}$ represent all edges of the directed graph. To make it more readable, the notation $A \rightarrow B$ is used to represent the transition function $\delta(A, Bt) = B$ where A and B represent a state and At represents any POI triggered by B. All transition functions that are not mentioned will have no effect and can be defined as $\delta(Y, Xt) = Y$. $q0 = \{S\}$ represent all nodes in the directed graph, which are start nodes. Furthermore, $F = \{F\}$ represents all nodes in the directed graph that are finished nodes.

Starting points are defined as all points of interest triggered by a start node. Figure 10 shows the POIs of the start node in the biathlon example. The presented Algorithm 1 describes this step.

Algorithm 1 Searching for solutions starts at each start node.

```
solutions ← []
startNodes ← [node | node.start == True]
for all s ∈ startNodes do
    poi ← fetchPointsOfInterestForNode(s)
    for all p ∈ poi do
        sol ← findPartialSolution(s, p)
        solutions.append(sol)
    end for
end for
return solutions
```

The DFA runs until it hits an accepting state, leading to "a partial solution". A partial solution consists of the taken paths and timestamps and describes one detected movement. It is defined as only partial to ensure the detection of all movements in the dataset. The final or total solution will be built in the algorithm's last step and constructed from multiple partial solutions.

After the automaton reaches an accepting state, the path taken and timestamps will be logged as a partial solution and then returned. The operation is described in Algorithm 2. Currently, each POI of a specific node is used as input and attempts to find a partial solution; however, there are specific cases where certain restrictions need to be applied.

In our example, let's consider the gate labeled as "S" (start). As shown in the map view of Figure 12, the athlete crossed the virtual start gate S several times before the race started. This might happen during the warm-up phase or the course inspection. Our approach would take those false triggers and build a valid solution. To avoid this scenario, users might specify a minimum or maximum duration for an edge. In this case, a maximum 60-second restriction is set for the path from

Algorithm 2 The search for solutions starts at every starting node.

Function FINDPARTIALSOLUTION($start, point, solution$)

```
solutions ← []
startNodes ← [node | node.start == True]
for all s ∈ startNodes do
  poi ← fetchPointsOfInterestForNode(s)
  for all p ∈ poi do
    sol ← findPartialSolution(s, p)
    solutions.append(sol)
    FINDPARTIALSOLUTION(start, point, solution)
  end for
end for
if len(solutions) == 0 and len(solution) > 0 then
  if lastNodeInSolution.finish == True then
    return [solution]
  end if
end if
return solutions
```

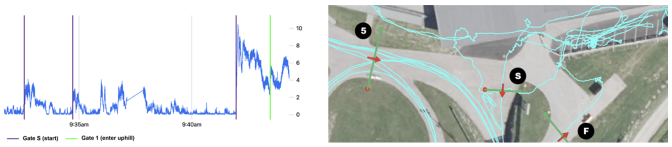

Fig. 12: Start gate "S" was triggered multiple times. In this case, only the last start trigger is valid.

gate S (start) to gate 1 (enter uphill). It is a method to narrow down the solution space to only valid solutions according to training or race rules, reducing space and time complexity.

E. Combine partial solutions to find multiple total solutions

Only connected ones are found in the search for partial solutions, indicating a valid POI sequence followed by another sequence. Imagine a break between two sequences: One partial solution will be found for the first sequence and another for the second sequence.

All partial solutions should be combined in all possible valid ways. "Valid" means that overlapping sequences cannot

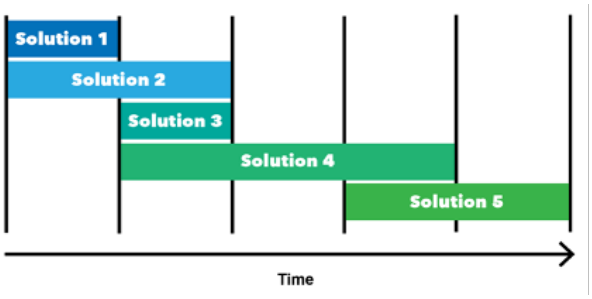

Fig. 13: Five partial solutions displayed by their temporal coverage. Note that each color represents a different partial solution.

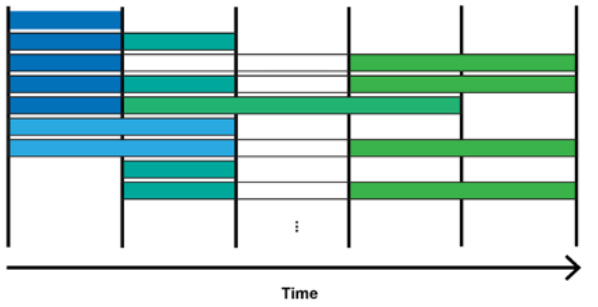

Fig. 14: Combining partial solutions to form valid total solutions.

happen simultaneously. Temporal order must be preserved, and solutions cannot be moved on the time axis. A list of solutions will be obtained, while a single solution may consist of one or multiple partial solutions. Each partial solution is also considered a valid solution.

F. Find optimal total solution

After identifying all potential solutions, establishing a metric for comparing and ranking them is essential. The specific metric will vary depending on the particular use case, but reasonable assumptions can be made to strike a balance between different approaches. Let's examine some of these approaches:

1) Maximize number of partial solutions: The most obvious would be maximizing the number of partial solutions in one solution. Let p_i be the partial solution of s with index i and n, which is the partial solution in s. The solution s is ranked by:

$$rankBy(s) = \sum_{i=0}^{n} 1$$

The disadvantage would be that many short solutions would be ranked at the top. This leads to a fragmented solution shown in Figure 15.

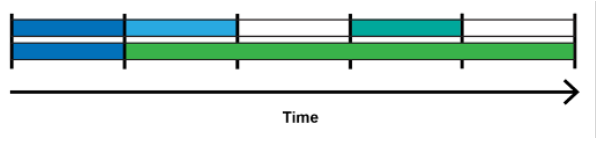

Fig. 15: Fragmented solution vs. de-fragmented solution.

Fragmented solutions tend to lead to wrong solutions as false triggers in Section III-C are promoted. This is because maximizing the number of partial solutions minimizes the duration covered by each partial solution. The algorithm in Section III-D produces partial solutions with short coverage (i.e., especially when false triggers occur). In general, there is no way of determining the "correct" trigger. It might be the first one, but it could also be the second one. Suppose multiple solutions have the same number of partial solutions. In that

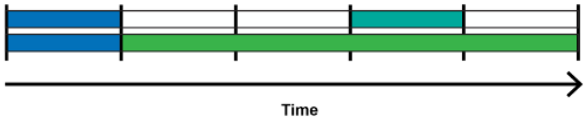

Fig. 16: Two solutions with each two partial solutions but different total duration.

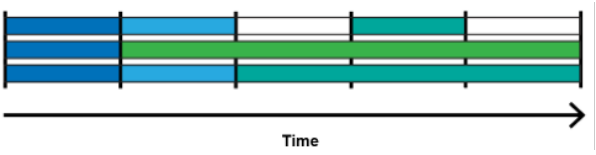

Fig. 17: Last solution will be taken.

case, the result will be an undefined behavior as it's impossible to determine the "best" solution (as shown in Figure 16).

2) Maximize the covered duration: To mitigate a fragmented solution, the total duration is maximized and covered in the solution. The total duration of a solution is the sum of all durations of partial solutions. Let $duration$ be a function to calculate the duration (end time – start time), and then the total duration is the sum of it. The solution s is ranked by:

$$rankBy(s) = \sum_{i=0}^{n} duration(p_i)$$

3) Combination: Both approaches will inevitably lead to non-deterministic behavior as there will be a lot of solutions with the same number of partial solutions or covered duration. The best approach is to combine both methods. First, rank for the maximum covered duration and then maximize the number of partial solutions(as illustrated in Figure 17).

IV. Evaluation

A. Implementation, Testbed, and Dataset

In this paper, sample data was collected in biathlon by a wearable IMU sensor and GNSS receiver. The sensor "OCULUS" is made by the Austrian company Lympik[1], a sports technology company focused on professional sports analytics. The accelerometer was configured at 50Hz and capable of measuring up to 16G (i.e., gravitational force). The gyroscope was also configured at 50Hz and an upper limit of 2000 degrees per second. The GNSS receiver was configured to record at 10Hz in multi-constellation (i.e., GPS, Galileo, GLONASS) mode, and Satellite-based Augmentation Systems (SBAS) were also enabled.

Our test subjects were three professional athletes. The data was recorded in the summer when the biathlon was simulated on special rollers. The wearable was mounted on the athlete with a special shirt while the rifle was carried above the sensor to avoid disturbing it while taking it off and picking it up.

[1]Lympik, (https://www.lympik.com)

Fig. 18: OCULUS tracking device mounted on biathlon athlete.

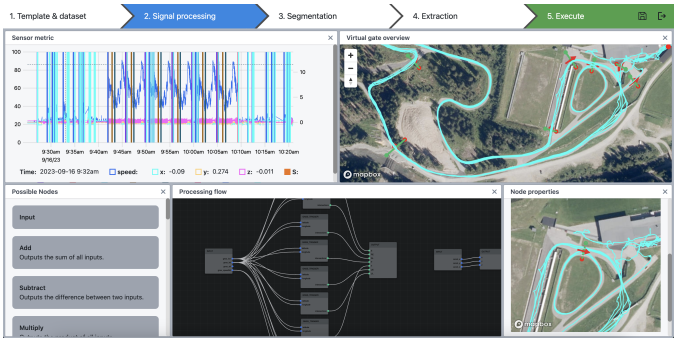

Fig. 19: Lympik sensor studio for analyzing sensor data.

Data analytics and visualization were done in Lympik Sensor Studio[1], a software service that quickly analyzes multi-sensor data. All functions used in the Studio are easily reproducible with simple scripts. The test implementation was done using Python. The raw binary sensor data was converted to standard units and signal processing was done with Scipy [2]. The location-based trigger is based on a simple line intersection in an Euclidean system.

For the detection of POI in our datasets, two kinds of triggers are used: Edge detection based on speed (i.e., the threshold was set to 1m/s) for detecting shooting start/finish and location-based triggers, which were positioned as shown in Figure 8. The domain-specific knowledge was encoded into a graph as presented in Figure 4. The resulting POIs were applied to our DFA as described in Section III-D. The path taken was recorded, and each edge was color-coded to recognize each segment easily. The segmentation is shown in a bar chart where the x-axis represents the time axis.

B. Experiments and Results

Our experiments were done with three biathlon athletes. This experiment aimed to test our approach to handling large datasets. Each dataset contains around 200k GNSS data points and 500k IMU data points[3]. The IMU of one tracker was configured at 200hz to test different dataset sizes as well. The knowledge of the track was obtained from a professional

[2]https://scipy.org/

[3]https://doi.org/10.5281/zenodo.13208678

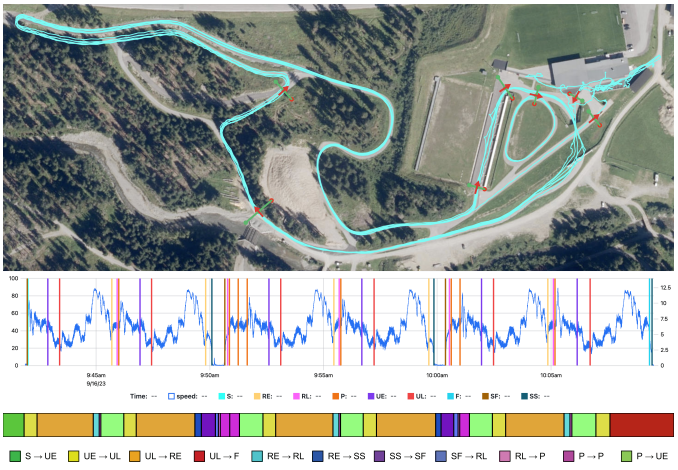

Fig. 20: Athlete A - dataset 191.9k GNSS data points, 503.1k IMU data points.

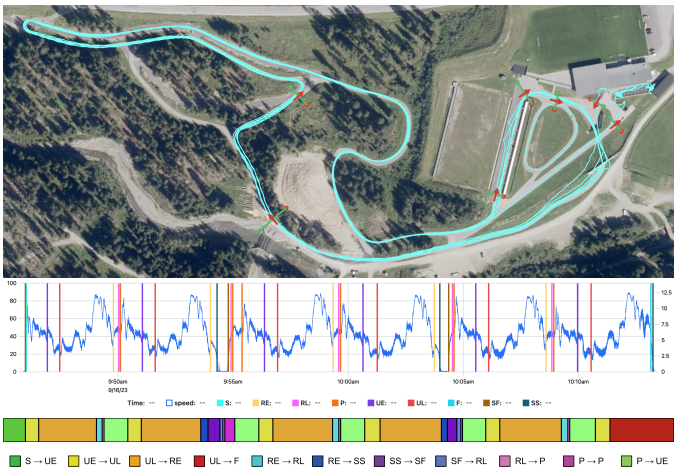

Fig. 21: Athlete B - dataset 197.6k GNSS data points, 493.7k IMU data points.

trainer and encoded into the graph as described in Section III-B.

In total, each athlete did six laps. Shooting was done only every 2nd lap; so, the athletes passed the shooting range without shooting in the 1st, 3rd and 5th. In the 6th and final lap, the athletes also skipped the shooting range and went to the finish. The raw GNSS data and virtual gates are visualized on a map. The chart displays the POIs triggered by each virtual gate, and the bar chart presents the optimal solution generated by the algorithm.

All three datasets were correctly segmented and found the optimal solution. Based on each segment, further calculations were made, such as calculating the average speed or maximum forces applied to the athlete. In dataset A (see Figure 20), the bar chart clearly shows the number of penalty rounds the athlete takes in pink $P \rightarrow P$. In the first shooting, the athlete missed two shots and took two penalty rounds, and in the 2nd shooting missed one. Athlete B, shown in Figure 21, did one

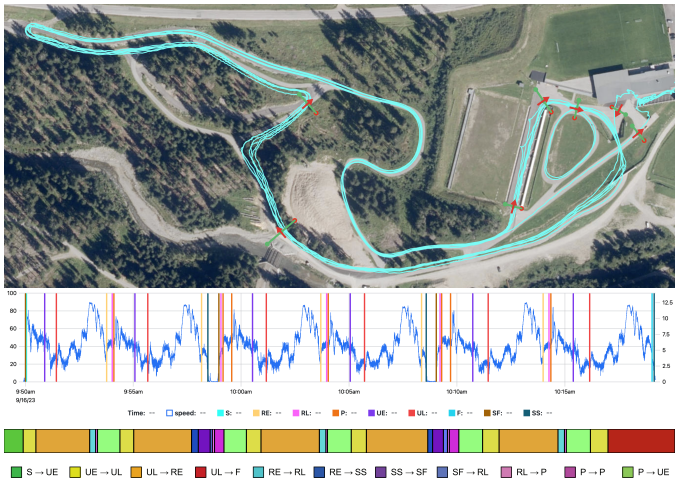

Fig. 22: Athlete C - dataset 205.5k GNSS data points, 2.1M IMU data points.

Dataset	Lap	Range time	Shooting time	Shooting Z-accel.
Athlete A	2	57.53s	33.90s	0.33G
Athlete A	4	54.49s	30.80s	-0.03G
Athlete B	2	53.47s	29.00s	0.27G
Athlete B	4	46.85s	22.90s	-0.07G
Athlete C	2	53.67s	30.60s	0.49G
Athlete C	4	51.11s	28.10s	0.18G

TABLE I: Duration and acceleration measured in shooting range.

penalty round in the first shooting and none in the 2nd. Results from Athlete C in Figure 22 show that the athlete missed one shot each time.

Upon closer examination, the numerical data in Table I reveals an interesting observation. All athletes show a faster range time ($RE \rightarrow SS \rightarrow SF \rightarrow RL$) in the 4th lap than the 2nd. Also, the Z-acceleration while shooting is different. The easy explanation for this correlation is that in the 2nd lap, athletes were told to shoot in a lying position while in the 4th, they had to stand. Standing results in a faster range time overall and different acceleration information.

Another experiment was to test the penalty round, which marks a special case. In this case, the athlete crosses multiple times (i.e., depending on the missed shots) the same line, and the segmentation still needs to work correctly (i.e., the penalty gate trigger in orange in the line chart). As shown in Figure 23, the three vertical lines indicate that the athlete crossed this gate three times. One can see the penalty was also correctly segmented (i.e., the bar chart in pink). The bar chart segmentation aligns perfectly with the line chart representing the POIs in this example taken from athlete A's dataset.

C. Discussion, Limitations, and Future Work

All results presented in Section IV were calculated on datasets with a few hundred thousand data points within 1-2 milliseconds on a one vCPU machine on the Google Cloud Platform with 2GiB of memory. Compared to other solutions utilizing neural networks (i.e., examples mentioned

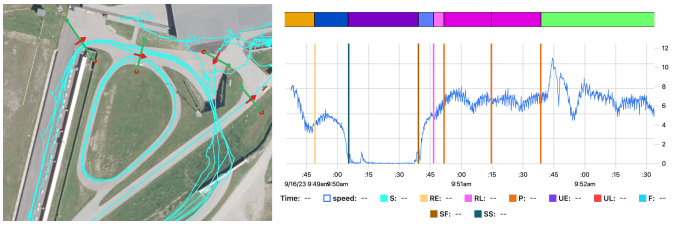

Fig. 23: A detailed view of the penalty round.

in the introduction), this method uses well-optimized and highly efficient analytic approaches combined with domain-specific knowledge to detect complex movements in a fast and resource-efficient way. There was no significant difference in processing time between datasets (i.e., dataset C had four times more IMU data than the others), further solidifying our approach for big datasets. Furthermore, no big datasets for training are needed, and traceability is a further advantage of the proposed approach.

The limitation of the proposed approach is the required domain-specific knowledge. One needs to know how a complex movement can be split into points of interest. A further challenge is posed by movement sequences, in which athletes do not perform the same way every time. In the future, we plan to parallelize the search for sub-solutions in Section III-D. One potential area to explore is the potential to remove the necessity for domain-specific knowledge. This could be achieved by utilizing a neural network or federated learning concepts (i.e., as presented in [25]) to embody domain-specific insights from extensive datasets and construct a graph based on the acquired knowledge. Future work remains investigating possibilities for real-time insights while enabling data processing in a distributed manner in the computing continuum [26], [27]. Lastly, we will explore integrating advanced sensor data analytics with AI planning techniques [28] in edge computing environments [29] to enhance the real-time performance and scalability of physical activity monitoring systems.

V. Conclusion

Integrating advanced wearable sensor devices has revolutionized capturing detailed movement data during physical activities. Such capability provides invaluable insights into performance metrics, learning fatigue points, and tracking movement efficiency. The proposed approach leverages sensor data and, via graph-based, captures and analyzes detailed movement data from individuals during various physical activities. This methodology offers a comprehensive solution for identifying critical performance metrics and fatigue points. Visualizing these data through graphs enables a clear and intuitive understanding of where and why performance drops occur. Nevertheless, this paper outlines just the initial stage of operationalizing the framework. In future work, we aim to develop a comprehensive technical framework and provide a thorough evaluation.

Acknowledgment

This work has been partially supported by the European Union's Horizon Europe research and innovation program under grant agreements No. 101135576 (INTEND), No. 101079214 (AIoTwin), and No. 101070186 (TEADAL).

References

[1] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang, "Solving rubik's cube with a robot hand," *CoRR*, vol. abs/1910.07113, 2019.

[2] G. V. Research, "Wearable technology market size, share & trends analysis report by product (head & eyewear, wristwear), by application (consumer electronics, healthcare), by region (asia pacific, europe), and segment forecasts, 2023 - 2030." https://www.grandviewresearch.com/industry-analysis/wearable-technology-market, 2023.

[3] M.-Z. Poh, K. Kim, A. Goessling, N. Swenson, and R. Picard, "Cardiovascular monitoring using earphones and a mobile device," *Pervasive Computing, IEEE*, vol. 11, pp. 1 – 1, 01 2011.

[4] P. Salvo, F. Di Francesco, D. Costanzo, C. Ferrari, M. Trivella, and D. de rossi, "A wearable sensor for measuring sweat rate," *Sensors Journal, IEEE*, vol. 10, pp. 1557 – 1558, 11 2010.

[5] C. Yu, Z. Xu, K. Yan, Y.-R. Chien, S.-H. Fang, and H.-C. Wu, "Noninvasive human activity recognition using millimeter-wave radar," *IEEE Systems Journal*, vol. 16, pp. 1–12, 06 2022.

[6] O. D. Lara and M. A. Labrador, "A survey on human activity recognition using wearable sensors," *IEEE communications surveys & tutorials*, vol. 15, no. 3, pp. 1192–1209, 2012.

[7] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," vol. 35, pp. 495–502, 08 2010.

[8] V. Bianchi, M. Bassoli, G. Lombardo, P. Fornacciari, M. Mordonini, and I. De Munari, "Iot wearable sensor and deep learning: An integrated approach for personalized human activity recognition in a smart home environment," *IEEE Internet of Things Journal*, vol. PP, pp. 1–1, 05 2019.

[9] O. Dehzangi and V. Sahu, "Imu-based robust human activity recognition using feature analysis, extraction, and reduction," pp. 1402–1407, 08 2018.

[10] J. Li, A. Sun, J. Han, and C. Li, "A survey on deep learning for named entity recognition," *IEEE transactions on knowledge and data engineering*, vol. 34, no. 1, pp. 50–70, 2020.

[11] A. Jalal, Y. Kim, and D. Kim, "Ridge body parts features for human pose estimation and recognition from rgb-d video data," in *Fifth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, pp. 1–6, 2014.

[12] T. F. N. Bukht, H. Rahman, M. Shaheen, A. Algarni, N. A. Almujally, and A. Jalal, "A review of video-based human activity recognition: theory, methods and applications," *Multimedia Tools and Applications*, pp. 1–47, 2024.

[13] P. Kumar, S. Chauhan, and L. K. Awasthi, "Human activity recognition (har) using deep learning: Review, methodologies, progress and future research directions," *Archives of Computational Methods in Engineering*, vol. 31, no. 1, pp. 179–219, 2024.

[14] Y. Kaya and E. K. Topuz, "Human activity recognition from multiple sensors data using deep cnns," *Multimedia Tools and Applications*, vol. 83, no. 4, pp. 10815–10838, 2024.

[15] J. Liu, D. J. Spakowicz, G. I. Ash, R. Hoyd, R. Ahluwalia, A. Zhang, S. Lou, D. Lee, J. Zhang, C. Presley, *et al.*, "Bayesian structural time series for biomedical sensor data: A flexible modeling framework for evaluating interventions," *PLoS computational biology*, vol. 17, no. 8, p. e1009303, 2021.

[16] P. Kumar and S. Suresh, "Deep-har: an ensemble deep learning model for recognizing the simple, complex, and heterogeneous human activities," *Multimedia Tools and Applications*, vol. 82, no. 20, pp. 30435–30462, 2023.

[17] Y. Zhang, X. Wang, J. Wen, and X. Zhu, "Wifi-based non-contact human presence detection technology," *Scientific Reports*, vol. 14, 02 2024.

[18] E. Ramanujam, T. Perumal, and S. Padmavathi, "Human activity recognition with smartphone and wearable sensors using deep learning techniques: A review," *IEEE Sensors Journal*, vol. 21, no. 12, pp. 13029–13040, 2021.

[19] D. B. West, *Introduction to graph theory*. Upper Saddle River, NJ: Pearson, 2 ed., Aug. 2000.
[20] F. Scholkmann, J. Boss, and M. Wolf, "An efficient algorithm for automatic peak detection in noisy periodic and quasi-periodic signals," *Algorithms*, vol. 5, no. 4, pp. 588–603, 2012.
[21] T. Koka and M. Muma, "Fast and sample accurate r-peak detection for noisy ecg using visibility graphs," in *2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pp. 121–126, 2022.
[22] F. Fell and M. Tanenbaum, "Preliminary comparisons of the wgs84(egm 96) geoid with national vertical datums," in *MTS/IEEE Oceans 2001. An Ocean Odyssey. Conference Proceedings (IEEE Cat. No.01CH37295)*, vol. 1, pp. 571–574 vol.1, 2001.
[23] G. Van Brummelen, *Heavenly mathematics*. Princeton, NJ: Princeton University Press, Apr. 2017.
[24] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to automata theory, languages, and computation*. Upper Saddle River, NJ: Pearson, 2 ed., Nov. 2000.
[25] I. Murturi, P. K. Donta, and S. Dustdar, "Community ai: Towards community-based federated learning," in *2023 IEEE 5th International Conference on Cognitive Machine Intelligence (CogMI)*, pp. 1–9, IEEE, 2023.
[26] P. K. Donta, I. Murturi, V. Casamayor Pujol, B. Sedlak, and S. Dustdar, "Exploring the potential of distributed computing continuum systems," *Computers*, vol. 12, no. 10, p. 198, 2023.
[27] V. Casamayor Pujol, A. Morichetta, I. Murturi, P. Kumar Donta, and S. Dustdar, "Fundamental research challenges for distributed computing continuum systems," *Information*, vol. 14, no. 3, 2023.
[28] I. Murturi, A. Egyed, and S. Dustdar, "Utilizing ai planning on the edge," *IEEE Internet Computing*, vol. 26, no. 2, pp. 28–35, 2022.
[29] I. Murturi and S. Dustdar, "Decent: A decentralized configurator for controlling elasticity in dynamic edge networks," *ACM Transactions on Internet Technology (TOIT)*, 2022.