Demonstration of a Stream Reasoning Platform on Low-End Devices to Enable Personalized Real-Time Cycling Feedback

Mathias De Brouwer[1,2][0000−0001−8769−6861], Femke Ongenae[1][0000−0003−2529−5477], and Filip De Turck[1][0000−0003−4824−1199]

[1] Ghent University – imec, IDLab, Department of Information Technology
iGent Tower, Technologiepark-Zwijnaarde 126, B-9052 Ghent, Belgium
[2] mrdbrouw.DeBrouwer@UGent.be

Abstract During amateur cycling training, analyzing sensor data in real-time would allow riders to receive immediate feedback on how they are performing, and adapt their training accordingly. In this paper, a solution with Semantic Web technologies is presented that gives such real-time personalized feedback, by integrating the data streams with domain knowledge, rider profiles & other context data. This solution consists of a stream reasoning engine running on a low-end Raspberry Pi device, and a tablet app showing feedback based on the continuous query results. To demonstrate this in a static environment, a virtual training app is presented, allowing a user to simulate an amateur cycling training.

Keywords: Stream reasoning · Low-end devices · Real-time feedback · Personalization · Cycling.

1 Introduction

In recent years, the importance of using data in sports has significantly increased, both on a professional and amateur level. In cycling, riders can be equipped with various sensors, e.g., heart rate monitors, GPS sensors, speed sensors etc. Analyzing the data measured by these sensors in real-time allows a rider to receive immediate feedback on how he/she is performing. Especially during training, this would allow a rider to adapt his/her training in real-time according to a prescribed training plan. Personalization of this feedback is required to maximize its value. For example, the physiological profile of a rider includes the boundaries between the different heart rate training zones in cycling, which are an important aspect of real-time training feedback. These rider profiles, potentially complemented with other context data such as route information, should be integrated with domain knowledge and the sensor data streams.

In amateur cycling, resources are limited compared to professional cycling. Existing real-time sports analytics solutions used by amateur cyclists, such as Strava[3] and TrainingPeaks[4], only focus on post-processing the sensor data, and

[3] https://www.strava.com
[4] https://www.trainingpeaks.com

Figure 1: Architecture of the real-time training feedback platform

do not focus on giving real-time personalized feedback. Moreover, they do not allow to integrate with various heterogeneous sensors and background data sources, e.g., domain knowledge and rider profiles [3]. By using Semantic Web technologies, this can be made possible: sensor & background data can be semantically enriched using ontologies, and stream reasoning techniques can be applied that combine semantic reasoners with stream processing techniques to consolidate & analyze the sensor data streams & available background knowledge [4]. By running these techniques on an inexpensive low-end device, e.g., a Raspberry Pi, mobile phone or tablet, such a solution can also be adopted by amateur cyclists.

In this paper, a solution is presented that gives personalized real-time cycling feedback to amateurs during training, by continuously evaluating queries on a stream reasoning engine running on a Raspberry Pi. By using Semantic Web technologies, this solution enables the real-time integration and processing of heterogeneous sensor data streams and background knowledge & context data on a low-end device. As a demonstrator of the solution, a virtual cycling training app is presented. It is virtual as it allows a user to simulate an amateur cycling training from within a static environment.

2 Platform Architecture

The architecture of the training feedback platform, which is designed to give individual feedback to a rider, is visualized in Figure 1. It consists of three main components, which should all be mounted on the rider's bike: the sensors, the Raspberry Pi running the stream reasoning server, and a mobile phone or tablet with a feedback visualization app. Various heterogeneous sensor devices can be plugged in into the platform and communicate with the stream reasoning server over wireless technologies, by using an existing IoT platform designed in previous research [2]. The stream reasoning server [3] consists of a running instance of the C-SPARQL RDF stream processing engine [1]. A cycling ontology has been designed to model domain knowledge, rider profiles, other context data and sensor observations. This data is hosted on the stream reasoning server, and used by the C-SPARQL engine as input data. Depending on the desired feedback, different continuous queries can be registered to the engine, which publish their results on a WebSocket. A feedback visualization app can listen over Wi-Fi to this WebSocket, to visualize the query results as real-time training feedback for the rider. This can help the rider to instantly react to his/her performance.

```
SELECT ?uuid ?firstName ?lastName ?time ?tzName
FROM STREAM <http://idlab.ugent.be/cycling/stream> [RANGE 5s STEP 1s]
FROM <http://localhost:8177/cycling-riders.rdf>
FROM <http://localhost:8177/cycling-sensors.rdf>
WHERE {
  ?o sosa:hasResult ?ov . ?o sosa:resultTime ?time .
  ?ov rdf:type cycling-sosa:HeartRateObservationValue .
  ?ov schema:value ?heartRate .
  { SELECT ?sensor (MAX(f:timestamp(?x, sosa:madeBySensor, ?sensor)) AS ?ts)
    WHERE { ?x sosa:madeBySensor ?sensor . ?x sosa:hasResult ?xv .
            ?xv rdf:type cycling-sosa:HeartRateObservationValue . }
    GROUP BY ?sensor }
  ?sensor sosa:isHostedBy ?device . ?device cycling-sosa:UUID ?uuid .
  ?athlete cycling-profile:monitoredBy ?device .
  ?athlete schema:givenName ?firstName. ?athlete schema:familyName ?lastName.

  FILTER ( f:timestamp (?o, sosa:madeBySensor, ?sensor) = ?ts )

  ?athlete cycling-profile:hasThreshold ?thLB , ?thUB .
  ?thLB cycling-profile:isLowerBoundOf ?tzAthlete .
  ?thUB cycling-profile:isUpperBoundOf ?tzAthlete .
  ?thLB schema:value ?thValueLB . ?thUB schema:value ?thValueUB .
  ?tzAthlete rdf:type ?tz . ?tz rdfs:label ?tzName .

  FILTER ( ?heartRate > ?thValueLB ) FILTER ( ?heartRate <= ?thValueUB )
}
```

Listing 1: `getTrainingZone` C-SPARQL query (prefixes are omitted)

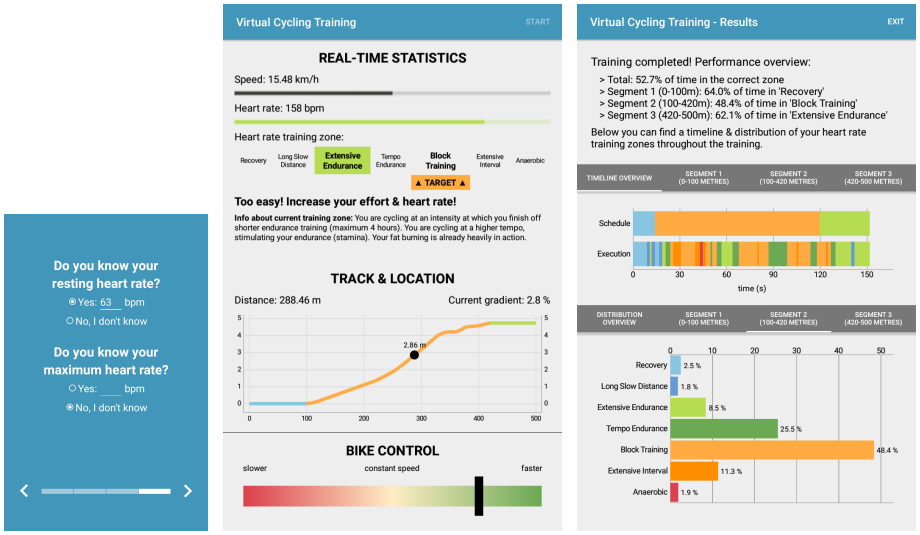
3 Use Case & Demonstrator: Virtual Training App

A virtual cycling training app, which is an adapted version of the visualization app in Figure 1, has been implemented for an Android tablet as a demonstrator of the proposed solution. The goal of the designed application is to let the demo user execute a cycling training. Due to portability and complexity issues, bringing a bike with a static bike system, e.g., Tacx, is unfeasible. Therefore, the cycling part is simulated by the user using the tablet's touch screen.

The architecture of the demo set-up consists of the Raspberry Pi and the tablet app of Figure 1. The sensors are replaced by virtual sensors in the app generating data based on the simulated cycling. On the Raspberry Pi, a C-SPARQL server is running. Two continuous queries are registered: `getQuantityObservationValue` to retrieve quantity sensor observations [3], and `getTrainingZone`, to retrieve the heart rate training zone corresponding to the rider's heart rate (Listing 1). Both queries are executed every 1 second on a window of 5 seconds. The C-SPARQL input data consists of the cycling ontology, the rider profile, the sensors' context data & the sensor data streams.

The tablet application consists of three different chronological parts, which can be categorized as pre-training, during training, and post-training.[5]

[5] The online demo page, including a video of the full demo process, is available at https://IBCNServices.github.io/cyclists-monitoring. The demo data (ontology, context data & queries) is also available at https://github.com/IBCNServices/cyclists-monitoring/tree/master/virtual-training-app-demo.



(a) Personalization (b) Virtual training (c) Data insights

Figure 2: Screenshots of the virtual cycling training app, chronologically with respect to the user executing the demo

Personalization (Pre-training). The first part of the app consists of a set of profile-related questions, which allow to construct the user's profile based on the answers and semantically represent it in the cycling ontology. In specific, the user's name, gender, birth date, fitness level, and resting and maximum heart rate (if known) are retrieved. An example of one of the app screens asking this data is shown in Figure 2a. Using expertise rules of thumb and the Karvonen formula, the user's personalized heart rate training zone boundaries can be determined from this data [3,5]. This enables to personalize training zone feedback for amateur cyclists, who cannot let their physiological profile be determined in expert lab tests, like in professional cycling.

Virtual Training. The second part of the app consists of the virtual training. The simulated use case involves the demo user being a rider (virtually) equipped with a heart rate monitor, riding on a bike (virtually) equipped with a speed sensor. The rider will execute a training on a specific track, where each track segment has a target heart rate training zone. The goal of the training is to ride in the target training zone as much as possible.

A screenshot of the real-time interface is presented in Figure 2b. This part consists of two panes: the upper pane (above the separator) consists of the real-time feedback and track & location data, while the lower pane contains the bike control UI. During a real-life training, the app will only consist of the upper pane; the control pane is only added to let the user control his/her speed.

As soon as the user starts the training, the user can use the slider in the control pane to increase his/her speed. The virtual sensors will start sending observations every 1 second to the C-SPARQL sensor streams. To generate these virtual sensor values, different algorithms are running in the app:

- The user's speed, initially 0, is iteratively increased or decreased with a value directly derived from the position of the slider in the control pane. If the gradient is non-zero, the speed is corrected with a factor depending on it. The speed is upper bounded based on the user's fitness level & the gradient.
- The user's heart rate starts in the recovery training zone. The next heart rate value is iteratively calculated based on the user's profile, speed, current heart rate, training zone, the gradient of the route, & a random factor.

Based on the results of the C-SPARQL queries, real-time feedback is shown to the user, in order to help him/her in achieving the training goals. The feedback includes the user's speed, heart rate, heart rate training zone, information on this zone, and an indicator whether to maintain, increase or decrease speed & effort. To allow for quick visual feedback during the cycling, a color is given to each training zone. The user can see a graphical representation of the track, as well as his/her location on it, to ensure he/she knows the current gradient of the route and the currently targeted training zone. The latter is also visually indicated.

Data Insights (Post-training). Using the data collected by the real-time processing, various data insights can be generated after the user has completed the full training. In particular, it is interesting to see how well the user has followed the prescribed training plan in terms of heart rate training zones. Therefore, the final part of the app shows a timeline and distribution of the user's training zones throughout the training. Both an overview and a dissection per segment are visualized. A screenshot of this part is shown in Figure 2c. Other insights could be generated using the data, e.g., by relating speed & heart rate over time.

Acknowledgements. F. Ongenae is funded by a UGent BOF postdoc grant. Part of this research was funded by the FWO SBO S004017N IDEAL-IoT and the imec.icon CONAMO, funded by imec, VLAIO, Rombit, Energy Lab & VRT.

References

1. Barbieri, D.F., et al.: C-SPARQL: a continuous query language for RDF data streams. International Journal of Semantic Computing **4**(1), 3–25 (2010)
2. Daneels, G., et al.: Real-time data dissemination and analytics platform for challenging IoT environments. In: GIIS 2017. pp. 23–30. IEEE (2017)
3. De Brouwer, M., et al.: Personalized real-time monitoring of amateur cyclists on low-end devices. In: WWW2018. pp. 1833–1840. ACM Press (2018)
4. Dell'Aglio, D., et al.: Stream reasoning: A survey and outlook. Data Science (Preprint), 1–25 (2017)
5. Kent, M.: Oxford dictionary of sports science and medicine, vol. 10. Oxford university press (2006)