David António Freire Moura

Bachelor in Computer Science

Exploratory Analysis of Individual Metrics in Team Sports Using *IoT* Sensors

Dissertation submitted in partial fulfillment
of the requirements for the degree of

Master of Science in
Computer Science and Engineering

Adviser:	Carmen Pires Morgado, Assistant Professor, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa
Co-adviser:	Ruben Duarte Dias da Costa, Director, Knowledge Biz Consulting

Examination Committee

Chair:	Name of the committee chairperson
Rapporteurs:	Name of a raporteur
	Name of another raporteur
Members:	Another member of the committee
	Yet another member of the committee

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

DRAFT: 2020-02-17

Exploratory Analysis of Individual Metrics in Team Sports Using Inertial Sensors

This document was created using the (pdf)LaTeX processor, based on the "novathesis" template[1], developed at the Dep. Informática of FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

Lorem ipsum.

Acknowledgements

The acknowledgements. You are free to write this section at your own will. However, usually it starts with the institutional acknowledgements (adviser, institution, grants, workmates, . . .) and then comes the personal acknowledgements (friends, family, . . .).

Abstract

Team Sports like basketball, football and baseball rely on data insights to improve player and team performance, practice scheduling and injury recovery. Recent improvements in data mining and machine learning techniques have encouraged the research of better ways to collect player data.

There are different approaches currently available, using video tracking systems or wearable sensors. The latter systems can be very precise in tracking position outdoors using GPS or indoors using Ultra-Wideband or RFID positioning, they fall short in the analysis of game related metrics, like accelerations and jumps or passes and shoots in the case of Basketball.

This work proposes a solution that can track both position and metrics of players and teams in real-time using inertial sensors, providing the users (e.g. players, coaches) an application with insightful information in order to improve the performance of the individual player and the whole team.

Resumo

Desportos colectivos tais como o basquetebol, futebol ou basebol usam dados para melhorar o desempenho em jogo, o planeamento de treinos e a recuperação de lesões de jogadores e da equipa. O aperfeiçoamento de técnicas como *data mining* e *machine learning* têm levado à pesquisa de formas de recolher mais e melhores dados dos jogadores.

Hoje em dia há várias abordagens diferentes: usando sistemas de captura de video ou utilizando sensores. Esta última abordagem pode ser bastante precisa no posicionamento em exteriores, utilizando sistemas como o GPS, ou em interiores, utilizando sistemas de posicionamento como Ultra-Wideband ou RFID. No entanto, estes sistemas não efetuam o reconhecimento de métricas de jogo como acelerações ou saltos, ou passes e lançamentos no basquetebol

Este trabalho propõe uma solução que visa medir em tempo real o posicionamento de jogadores e da equipa no campo, assim como recolher métricas de jogo, utilizando sensores de inércia. Deste modo, pretende-se fornecer aos utilizadores (e.g. jogadores, treinadores) uma aplicação com informações úteis e detalhadas sobre o desempenho dos jogadores individualmente, e da sua equipa como um todo.

Contents

List of Figures

List of Tables

Listings

Acronyms

API	Application Programming Interface
BLE	BLuetooth Low Energy
CRUD	Create, Read, Update and Delete
EPU	Edge Processing Unit
FIFA	Fédération Internationale de Football Association
GNSS	Global NAvigation Satellite system
GPS	Global Positioning system
IMU	Inertial Measurement Unit
IoT	Internet of Things
KBZ	Knowledgebiz Consulting
MEMS	Micro-electromechanical systems
NBA	National Basketball Association
NFC	Near Field Communication
NFL	National Footbal League
REST	Representational State Transfer
RFID	Radio-frequency identification
UWB	Ultra-wideband

Symbols

Chapter 1

Introduction

This chapter introduces this dissertation, presenting its motivation, the challenges it poses and the objectives to be achieved. After, the expected contribuitions are defined. It finishes by describing the organizational structure of the remainder of the document. This dissertation was developed within a corporate context, with the support of KnowledgeBiz Consulting (Knowledgebiz Consulting (KBZ)) [3].

1.1 Motivation and Context

In an era where information drives our world and big data analytics are a common interest in every academic and industrial field, sports industries like Basketball, Soccer and Baseball are using more and more data in their behalf. Huge volumes of data are produced per game and per training session, and techniques like data mining and machine learning can provide coaches and players with more insightful and accurate information and analysis [1].

New discoveries on data can also help the business side of sports, like the National Basketball Association (NBA) Drafts or the football transfer market.

Innovations like wearable Global Positioning system (GPS) tracking devices have added big insights in team sports. Initially, practitioners of sports were interested in answering the first obvious question of the chaotic set of player movements during a game: how far did an athlete traveled over the course of a match? GPS could easily answer this question, but soon more questions were raised. For example:

- How many high velocity efforts were achieved?
- How much time was spent at high speed?

- What high accelerations and decelerations were performed? What impact does this have on the recovery strategies for each individual athlete?

This granular movements can be measured in a laboratory, using biomechanical tools like force plates that measure ground reaction forces, but this measurement tools can't be deployed in real life cases like games or training sessions, and a "gross" positioning system like GPS can never measure this type of forces.

For a complete tracking solution, we need not only to analyze the "big picture" (the movement across the field), that can be provided by GPS, but to consider the micro or local movements, which can provide enormously relevant information for performance enhancing, fatigue analysis and recovery strategy planning.

Countless tracking systems exist for different sports, and even for different activities in each sport, but many of them are either very expensive, resource-intensive or attached to the field. Currently, there are two widely used systems to track the position and collect metrics from the players: sensors attached to each player, which have included a GPS unit and an inertial measuring unit (accelerometer, gyroscope and magnetometer), and an array of cameras placed around the field of the game (e.g. SportVU and recently Second Spectrum used in NBA) [5].

However, these systems have particular drawbacks: The GPS signal can accurately measure the position, the speed and the displacement of a player. But it can't measure metrics like jumps, shoots and passes or falls; When in a closed court (like most of basketball courts), there is no GPS signal. The tracking is made using cameras around the court; Visual systems can accurately track the position of the players and gather advanced metrics. The downside of these systems is the need of high processing power, and that the cameras are attached to the court. In this way, every court needs to have this expensive system installed to be able to track the players performance, something that is often out of reach for low ranked and amateur teams; Most times an operator is needed to tag the events recorded by the video systems, or the video is recorded and then the processed data is provided to the coaches and players.

In order to tackle the drawbacks of video and GPS solutions, Kinexon [2] developed a system using a wearable sensor that can accurately track basketball players positions indoors, using Ultra-WideBand (Ultra-wideband (UWB)) technology, and claims to monitor player load and efforts measured in real-time during practices or matches, providing tactic and technic analysis [4].

However, this system falls short in the field of micro-movement identification, where the analysis of shoots and passes or attacks and defenses can be a useful tool for coaches to improve the performance of the team.

With this motivation, KnowledgeBiz Consulting (KBZ) challenged me to develop a system capable of tracking the position and to collect insights on performance metrics of a single player and of the entire team, using real-time data from inertial sensors. Apart from the technical challenges, this system should also be available for every team, in any performance level, by being low-cost and easy to include in a training or match environment. KBZ is providing the devices that contain the inertial sensors (accelerometer, gyroscope and magnetometer), and communicate via Bluetooth.

1.2 Objectives

The proposed objective of this dissertation is to develop a system based on inertial sensors like accelerometers and gyroscopes, capable of tracking game metrics, in order to provide insights about player and team performance in real-time.

Several sports are often practiced indoors, which prevents the use of GPS positioning. In this cases, a tracking system using inertial sensors like accelerometers or gyroscopes can be used. In order to test this tracking system, basketball was the sport chosen.

Basketball is a good sport to track player movements because: it has a high number of events occurring in small time frames when comparing to other team sports like football; the number of players per team is 5 (a small number when compared to 7 players in a handball team or 11 players in a soccer team); it's one of the sports that invests more in new technologies (Internet of Things (IoT)) to track players and teams.

The first step of this dissertation will be a survey of current methods of data collection from basketball players, and how are the game metrics calculated from that data.

Afterwards, a system architecture that covers the data collection from multiple players, calculates game metrics from it, stores it for further use (history or statistical analysis) and presents it to the users should be designed.

Then, a prototype should be implemented. This prototype should collect data from multiple basketball players, from multiple parts of their body, to calculate different game metrics. These game metrics should be communicated to a central server (local or in the cloud), which should store all the information. A frontend application should be developed, displaying individual metrics for each player, but also providing an aggregated view regarding the whole team performance, like attacking and defending strategies in the form of heatmaps. Finally, this system (body sensor network and application) will be tested and validated, preferably in a real-life situation, like a game between two amateur teams.

Summarizing, the main goal of this dissertation is to study and analyse ways to track player positions and to measure basketball game metrics, collected from inertial sensors, and to illustrate those insights to the users (e.g. coaches, players) through an application, providing a central location to monitor and analyze the players and the team.

1.3 Expected Contributions

The expected contributions of this dissertation are:

1. Exploratory analysis of relevant basketball metrics, obtained from inertial sensor data
2. Implementation of a body sensor using a wireless communication protocol to transmit data
3. Development of an system that collects data from inertial sensors, analyses it and displays the insights for the players and the team
4. Publication of a paper in international conference proceedings

1.4 Document Structure

The rest of the document is organized as follows: Chapter 2 presents the related work. Chapter 3 will propose a system architecture that covers and solves the problems raised in this section. Chapter 4 will describe the implementation of a prototype developed during the elaboration of this dissertation. Chapter 5 will evaluate the results obtained with the developed prototype. Finishing this dissertation, Chapter 6 compares the developed work with the objectives set in the start of the dissertation, and analyses the final result. It also suggests future work.

CHAPTER 2

Related Work

Many studies carried in the past have focused on subjects related with this work, such as collecting human movement data, human activity recognition and the measuring of game metrics in sports.

The following will discuss the related works that served as a starting point to this dissertation.

Chapter 3

System Proposal

To achieve the goals set in Chapter 1, this chapter proposes a system architecture that collects and processes the raw data from the IMU sensors and transforms it into meaningful metrics, storing it, and then presents it in a clear way to the user.

3.1 Requirements

With the goals set, and after analyzing the current state of the art, the systems already being used in real life situations and their shortcomings, a set of requirements for the proposed system has to be defined.

It should be able to work indoors, thus avoiding the use of systems like GPS, and it should work in any field, without the need of preinstalled structures (like a video system).

It also should provide real-time data of individual game metrics, calculated with the raw data from the IMU Sensors.

In order to collect data from all the players of the game, the system should be scalable. The metrics should be calculated and sent to the user, wether collecting data from 1 player, 1 team or 2 teams (with additional substitute players).

3.2 Architecture

The proposed architecture is shown in Figure 3.1, and it is comprised of three segments: **Edge** - Data Collection and Processing; **Server** - Data Storing and Application Server; **Client** - Frontend Application.

The layering of the architecture allows that each segment is completely independent in its implementation. The architecture will be explained in more detail in the following subsections.

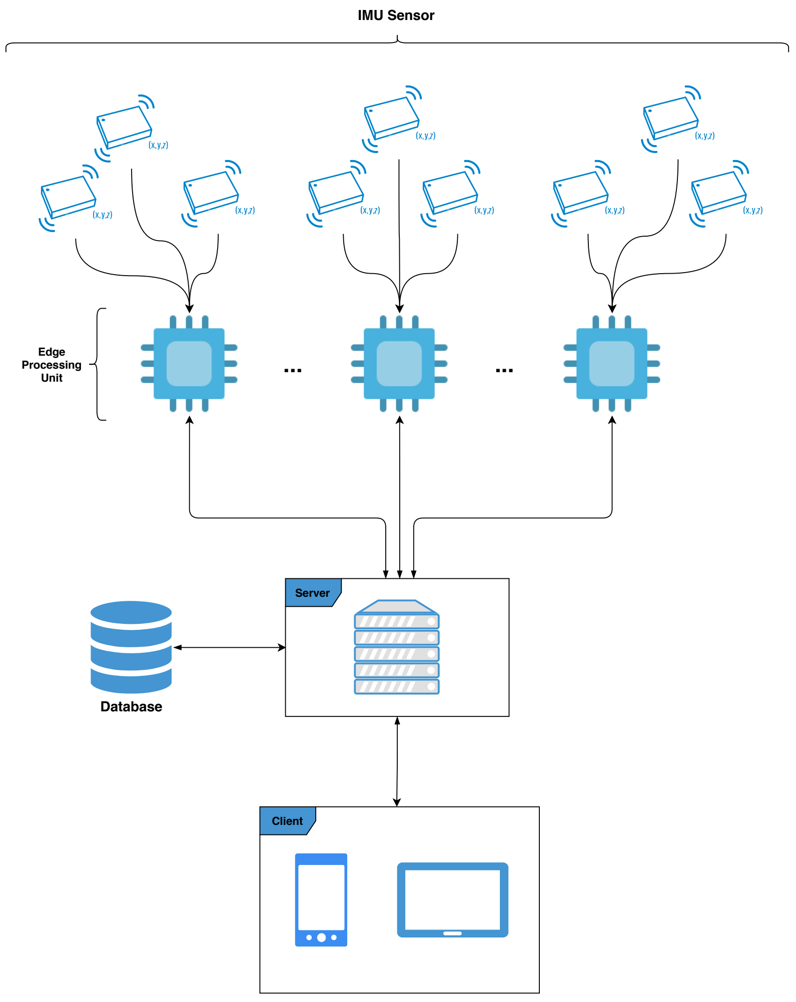


Figure 3.1: Proposed Architecture

3.2.1 Edge

The Edge segment is composed of at least one Inertial Measurement Unit (IMU) Sensor and one Edge Processing Unit (EPU), that controls and receives information directly from the IMU Sensor. The EPU should also be capable of calculating insightful metrics, calculated with the raw data acquired from the IMU Sensors.

3.2.1.1 IMU Sensors

A IMU Sensor is an integrated sensor package that measures a body's specific force, angular rate and orientation.

Specific force is a measurement of coordinate acceleration, which can be obtained by removing the gravitational acceleration. It is measured with accelerometers. Angular rate is the rate at which a body rotates, measured by gyroscopes. Orientation is a description of how a body is placed in the place it occupies. Using magnetometers, a body can be oriented relatively to the Earth's magnetic field.

For this application, IMU Sensors armed with 3-axis accelerometer and gyroscope should be used.

The IMU sensors should be attached to the player's body, in different locations (like the back, the foot or the hand), to track different metrics.

3.2.1.2 Edge Processing Unit

The EPU does the "hard work". Besides handling the connection and controlling the IMU Sensors operations, like starting and stopping the raw data collection, this unit has to collect the raw data sent by the IMU Sensors several times per second, and perform calculations with the gathered raw data to measure in-game metrics, which are then sent to the server, to be stored and shown to the User.

In order to be able to receive data and calculate in-game metrics from all the IMU Sensors, one EPU may not be sufficient. An array of EPU's, each one connected to different IMU Sensors and sending data in parallel to the Server should be used.

3.2.2 Server

The Server is the main piece of the architecture, allowing the interaction between the User and the IMU Sensors.

The Server can send instructions to the EPU, controlling the IMU Sensors operations. When data is received from the EPU, the Server stores it, so that it can be shown to the User, through the Client application.

The Server also hosts the Client Application, which communicates tightly with the Server.

3.2.3 Client

The Client Application provides the User an interface in which he can control and consult information about players, teams and games.

Indirectly he should trigger the start and stop of the data collection in the IMU Sensors, which will then send the raw data to the EPU. After the game metrics are calculated in the Edge, they are shown to the user, in the Client Interface.

Chapter 4

System Implementation

The description of the System Implementation according to the proposed architecture will be separated in the three segments: Edge, Server and Client. Figure 4.1 represents the implemented architecture.

4.1 Edge

As described in section 3.2.1, the edge segment is composed by one or more IMU Sensors and one or more EPU.

The IMU Sensors used in the implementation phase are armed with 3-axis accelerometer, which measures the specific force in m/s^2, 3-axis gyroscope, which measures the angular rate in $°/s$, and 3-axis magnetometer, which measures the strength of magnetic fields, in μT. The IMU sensors used also have a battery, internal storage, and communicate through Bluetooth Low Energy.

In this phase, a Raspberry Pi was chosen to have the role of EPU. A Raspberry Pi is a single-boarded, small and low-cost computer developed by the Raspberry Pi Foundation, with the goal of enabling people of all ages to explore computing. It was chosen for this system for its easy accessability in the market, reduced size, high portability and low price, its wide range of features, like Bluetooth and WiFi communications, and being a good platform for developing an application prototype.

The number of IMU sensors used can vary in each player by the number of metrics being measured (different metrics may require the use of more sensors) and the number of players being tracked.

As the number of sensors can grow, and due to the limitation of the number of connections to a Bluetooth receiver, the number of Raspberry Pi's can also grow, in order to establish connection with all the IMU Sensors. This also helps to share the computation

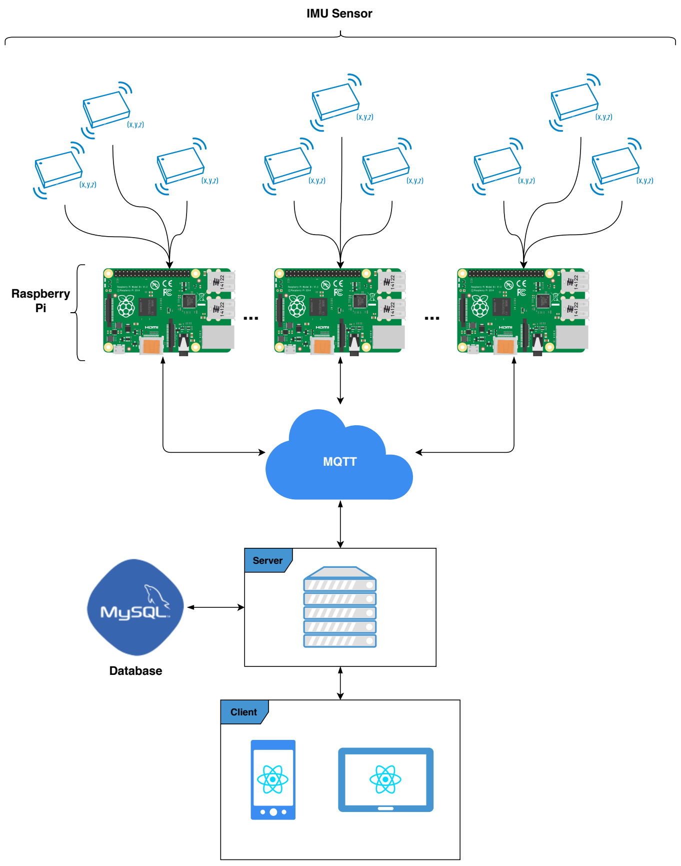


Figure 4.1: Implemented Architecture

of the game metrics between the Raspberry Pi's.

4.1.1 IMU Sensor Control and Communication

The role of the IMU Sensors is to receive instructions issued by the user, and send accelerometer and gyroscope raw data to the Raspberry Pi they are connected, over Bluetooth.

The following operations are supported by IMU Sensors:

- State control
  - Start Raw Data Collection
  - Stop Raw Data Collection
  - Shutdown
- Sample Rate
- MPU Configuration

Table 4.1: Raw Data Example

time	accX	accY	accZ	gyrX	gyrY	gyrZ
0.02	0,46178	-0,83982	9,46772	0,07084	0,05007	-0,03089
0.04	0,44264	-0,80991	9,42465	0,07031	0,04954	-0,03196
0.06	0,42948	-0,83384	9,42106	0,07031	0,04900	-0,03196
0.08	0,41991	-0,82786	9,44858	0,07031	0,04954	-0,03196
0.10	0,43666	-0,82666	9,41030	0,06978	0,04794	-0,03196

The IMU Sensors must pair with a Raspberry Pi, which is searching for the known sensors. After being paired, the operations can be communicated to the IMU Sensors through Bluetooth Notifications.

Before starting the data collection, it is necessary to set the sample rate to 50 Hz (necessary for the metrics algorithms), and to activate the gyroscope and the magnetometer sensor by changing the MPU Configuration. By default only the accelerometer sensor is active.

To start collecting data from an IMU Sensor, it is needed to provide information like the MAC Address of the sensor, an identification of the game being played and the player wearing the sensor (because the sensors can be changed from game to game between the players). It is also needed to send the position of the sensor in the player's body, to calculate the metrics, as each algorithm uses data collected from a different part of the body.

The IMU Sensors send time (ms), accelerometer ()m/s^2) and gyroscope ($°/s$) data in a bite array, which is then converted to human-readable values. The magnetometer data isn't used, so it is discarded. An example of raw data sent by the IMU Sensors can be seen in Table 4.1.

When the metrics are calculated, they are sent to the server via MQTT, to be stored there. The Raspberry Pi's don't store data, they only server as a vehicle of instructions and data between the server and the sensors.

A single Raspberry Pi is then responsible by maintaining the connection with several IMU Sensors; communicate instructions; receive accelerometer, gyroscope and magnetometer data, and stores it temporarily; depending on the position of the sensor, calculate the according metrics algorithms with the received data; send the calculated metrics to the Server.

4.1.2 Game Metrics

To achieve the goal to collect insightful information of a basketball game, a set of game metrics were chosen, focusing on three body locations: foot, lower back and hand.

Using the sensor in the foot we can infer the number of steps, traveled distance and player trajectory.

In the hand, it is possible to detect the number of dribbles, and time dribbling. With this, it is also possible to calculate the time of ball possession of the whole team.

In the lower back, the sensor was used to collect data and analyze it to detect the number of jumps and time in air.

The following sections will explain in detail how the metrics are calculated, using raw data.

4.1.2.1 Trajectories

4.1.2.2 Steps

The detection of steps takes advantage of the algorithm explained in 4.1.2.1, especially the Zero-Velocity Detection. Whenever a Zero-Velocity is detected, it is recorded in a binary array (zeros meaning no movement, ones meaning movement). Afterwards, when the Trajectory detection algorithm ends, the array is analysed.

The analysis is made by counting the meaningful "sections" of ones. Listing 4.1 shows an example of a step detection array, where we can clearly see there are two main sections" of ones, meaning that using this example the algorithm should count two steps.

Listing 4.1: Steps Array Example

```
[00000000001000011111111111111110001000000000000000101111111111111111111001100000000000]
```

However the algorithm must take into account the outliers, and shouldn't identify it as steps. This can be ensured by only counting a step after a number of consecutive ones is observed. If two consecutive zeros appears in the middle, the possible step is discarded. The behavior of the algorithm is described in Listing 4.2.

Listing 4.2: Steps Counting Algorithm

```
steps = 0;
no_ones = 0;
last = 0;
for idx = 1:data_size
   current = walking(idx);
   if (last == 0 && current == 1)
      no_ones++;
   elseif (last == 1 && current == 1)
      no_ones++;
      if (no_ones == 12)
         steps++;
      endif
   elseif (last == 0 && current == 0)
      no_ones = 0;
   endif
   last = current;
endfor
```

This algorithm can be improved in the detection of the "sections" of ones, as it can sometimes fail. For example, if in one step sequence we had 3 occurrences of zeros inside a large sequence of ones, the algorithm would count 2 steps instead of 1 step However,

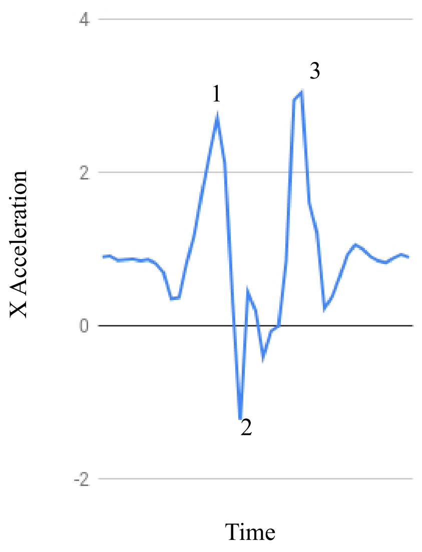


Figure 4.2: Jump X Acceleration

when testing the results were satisfactory, so no more work was done in the development of this algorithm.

4.1.2.3 Dribbles

4.1.2.4 Jumps

The method of identifying jumps was developed by Alexandre Martins, during his internship in KBZ. Alexandre's approach to jump detection relied on the recognition of a jumping pattern in the data collected from a IMU Sensor placed in the lower back.

To start this analysis, he first collected data from all the sensors of the IMU while jumping, and searched for a pattern that could be used to identify the jump movement. That pattern was found in the X axis of the accelerometer sensor (the vertical axis), and it is represented in figure 4.2.

As identified by the number in figure 4.2, a jump is composed by 3 moments: the feet leave the ground; maximum jump height reached; the feet touch the ground. These moments are easily identified by being a maximum value, followed by a minimum, followed by a maximum, inside a time window.

After identifying this pattern, Alexandre suggests a Fourier analysis on the data to verify it is a jump, and not a running movement, which are easily confusable. Compared to running, jumps are a slower movement, and will show a lower frequency in the Fourier analysis. After testing, Alexandre concluded that the usual frequency of jumping is 1.25Hz, as shown in figure 4.3.

The algorithm that was developed by Alexandre was used in the system implementation, and received the real time data from the X-axis of the IMU Sensor placed in the lower back.

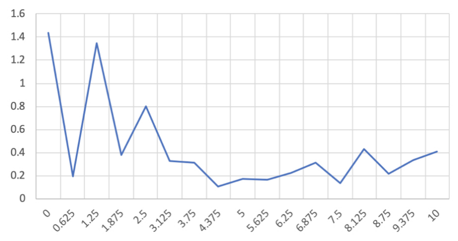


Figure 4.3: Jump Fourier Analysis

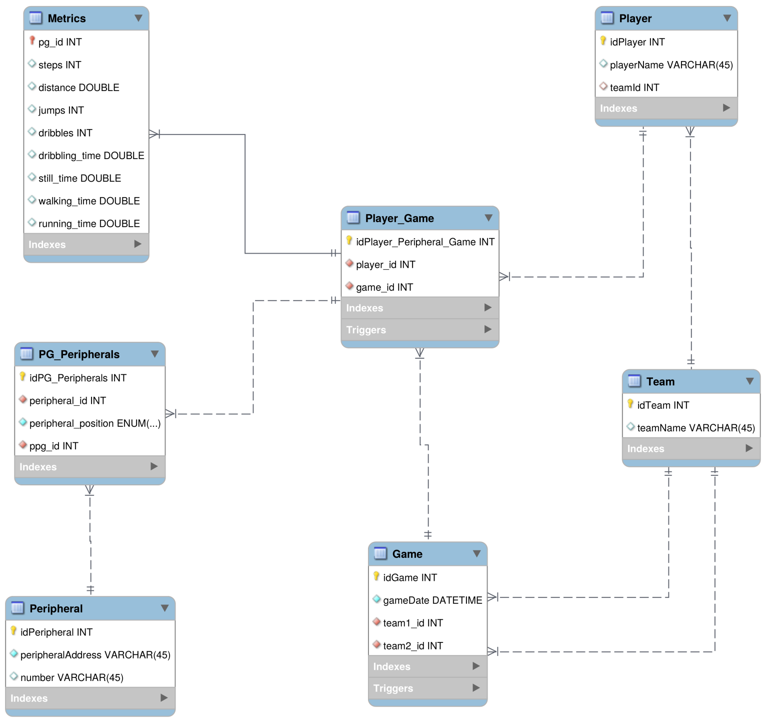


Figure 4.4: Database Model

4.2 Server

The server is built as a Representational State Transfer (REST) Application Programming Interface (API), developed with Node.js, and taking advantage of the Express framework that enables a fast development of Web Applications and API's.

To manage peripherals (IMU Sensors), Players, Teams, and Games, there are API endpoints for each one of this resources, and all the data is saved in a MySQL Database.

4.2.1 Database Model

The database, developed in MySQL, was designed in order to store all the data from the main resources, its relations and meaningful data in those relations. Figure 4.4 shows the database model, which will be explained in detail.

The main resources are represented in the following tables: Peripherals; Players;

Teams; Games; Metrics.

Each entry in the Peripheral table represents an IMU Sensor. It's stored the peripheral MAC Address, and its attributed number for easier identification.

In the Players table, each player has stored its identifier, name, and an identifier of the team they belong to.

The Team table only stores the name of the team and its identifier.

To store information about a game, the Game table has its identifier, the date of the game, and a reference for both teams attending the game.

For the metrics, each player has a set of metrics per game, with the data gathered from all the IMU Sensors he's using. The metrics being currently stored are: number of steps and distance traveled; number of jumps; number of dribbles and dribbling time; time spent being still, walking and running.

One of the features of the system is to be able to handle multiple peripherals per player, in different body positions, that can be changed between games. To store this information, two tables are needed: one that stores the relation between the player and the game (Player_Game Table), and another that relates that with a peripheral (PG_Peripheral Table), storing where the peripheral is placed in the player's body.

4.2.2 API Description

The Developed API exposes endpoints to control the different resources available in the system. Those resources are:

- Peripherals (IMU Sensors)
- Players
- Teams
- Games

4.2.2.1 Teams

The Teams endpoint enables the Create, Read, Update and Delete (CRUD) actions over the teams of the system.

It lets the user create a new team, given their name, the ability to consult all the teams registered in the system, to consult a specific team and their respective information or the players that belong to that team. This endpoint also serves to update information about a team or to delete the team from the system.

4.2.2.2 Players

The Players endpoint provides the user the CRUD actions over the players of the system.

It enables the creation of a new player, given their name and the team they belong to, the ability to consult all the players registered in the system, or to consult a specific player and their respective information; for a given player, this endpoint also serves to update information or to delete the player from the system.

4.2.2.3 Games

This endpoint, besides providing Create, Read and Delete operations for the games, also manages the attribution of peripherals to each player, in multiple parts of their body, and also to fetch the metrics for each player.

For the Create operation, it is possible to create a game providing the date of the game and the two teams that are playing the game.

For the Read operation, it is possible to get a list of the games (that already occurred or that will occur), get the main info about a specific game (the date and the teams) or to get specific info about a specific game (like the players of each team, and the assigned peripherals for each player).

For the Delete operation, it is possible to delete a game.

To manage the peripherals for each player, it is possible to assign a peripheral to a player in a given game, providing the player identifier, the peripheral address and the location of the body where the IMU Sensor is located, and it is possible to remove this assignment.

It is also possible to read the metrics for each player, and to get a sum of the dribbling time for each team, which gives an insight for the total ball possession during a game.

4.2.2.4 Peripherals

The peripherals endpoint enables the control of the IMU Sensors from the Client app, and stores temporary state on the peripherals, like what are the currently connected peripherals, and which ones are collecting data.

As specified in Subsection 4.1.1, there are 3 main controls that can be sent to the IMU Sensors: start, stop or shutdown. The Peripherals endpoint allows 3 ways of sending this command to the Edge segment: control only one given peripheral, control all the peripherals in a game or control all connected devices.

The control instructions are sent to the Edge Segment through the MQTT protocol, and are activated notifications to receive updates from the Edge. For example, if the instructions to start all the peripherals in a game, the server will be awaiting for notifications from each of the peripherals, with the calculated game metrics, depending on the position of the IMU Sensor in the body of the player.

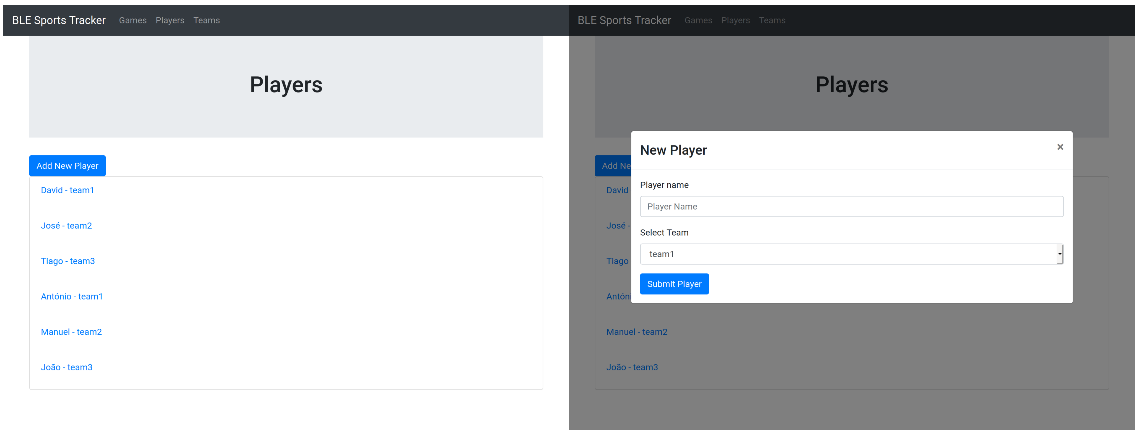
4.3 Client

The client application was developed in React, and is capable of controlling the IMU Sensors, and manage teams, players and games. It also displays to the user information about Player and Team game-related metrics.

The application communicates directly with the API, through HTTP, and consumes the API exposed by the server. This means that the Client application is divided into 3 parts: players, teams and games.

4.3.1 Players

The main Players page shows a list of all players, and their current team, as shown in Figure 4.5a. It also allows the user to add a player, by providing their name and the team they belong to, as shown in Figure 4.5b.



David - team1
José - team2
Tiago - team3
António - team1
Manuel - team2
João - team3

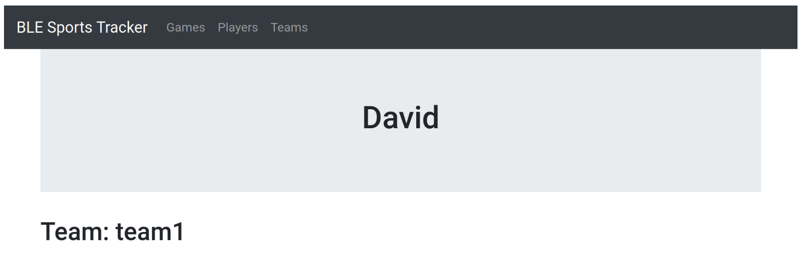
(a) Players List (b) Add Player

Figure 4.5: Players List Page

The player detail page shows the information about a single player, and supports the operations of editing the details of the player and deleting the player. In the current system implementation, shown in Figure 4.6 , only the information of the player is displayed, the other operations are currently unimplemented on the client application.

4.3.2 Teams

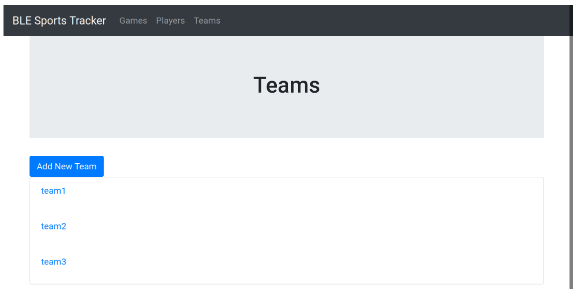
The Teams page shows a list of all the teams registered in the system, and allows the creation of new teams, given the team name. This pages are illustrated in Figure 4.7a and



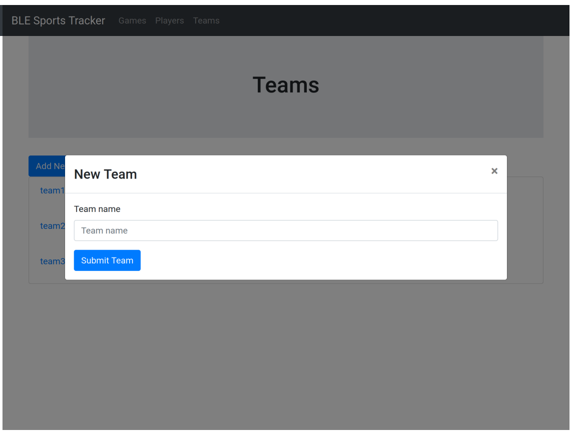
Team: team1

Figure 4.6: Players Detail Page

Figure 4.7b.



(a) Teams List



(b) Add Team

Figure 4.7: Teams List Page

The Team detail page shows the information about a team, and the team players. As explained in 4.2.2.1, the API supports the operations of editing and deleting teams. However, as shown in Figure 4.8, it was not implemented in this system.

4.3.3 Games

The Games page shows a list of past and future games, with the date of the game, and the two teams that are playing that game. In this pages, it is also possible to create a new game, providing the date and the opponent teams. This page is shown in Figure 4.9. Similar to the previous pages, even though it is supported by the API, it isn't possible to edit game details and delete a game.

In a game detail page (Figure 4.10), it is displayed more information about the game, like the players of each team, what IMU Sensors are they wearing and were they are located.

In this page the user can assign IMU Sensors to each player, as shown in Figure 4.11a, and can unassign them as well. They can also control all the sensors assigned to players

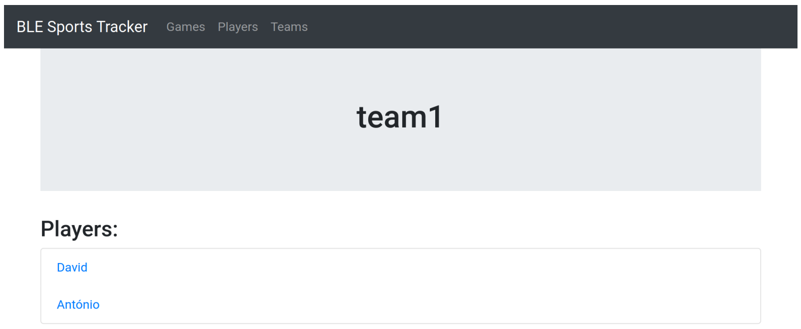
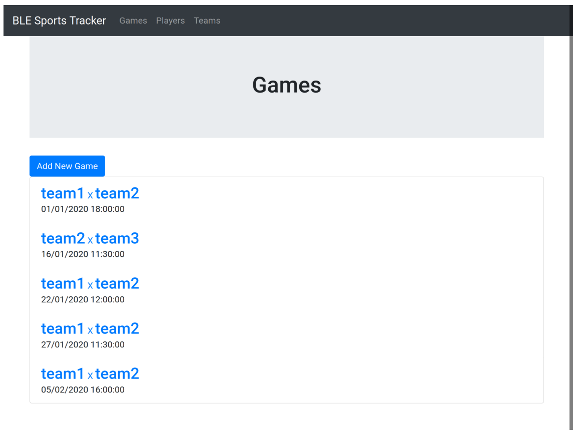


Figure 4.8: Teams Detail Page



Games

team1 x team2
01/01/2020 18:00:00

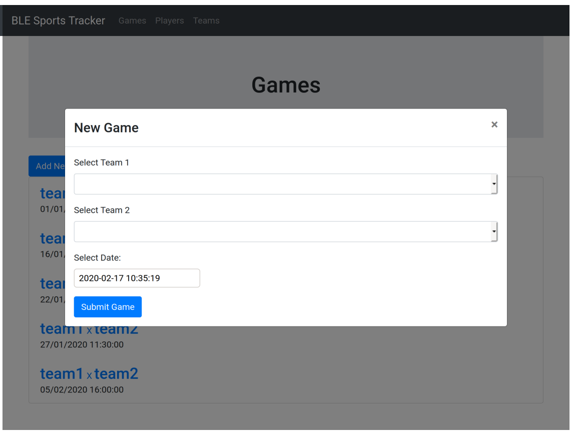
team2 x team3
16/01/2020 11:30:00

team1 x team2
22/01/2020 12:00:00

team1 x team2
27/01/2020 11:30:00

team1 x team2
05/02/2020 16:00:00

(a) Games List



Games

New Game

Select Team 1

Select Team 2

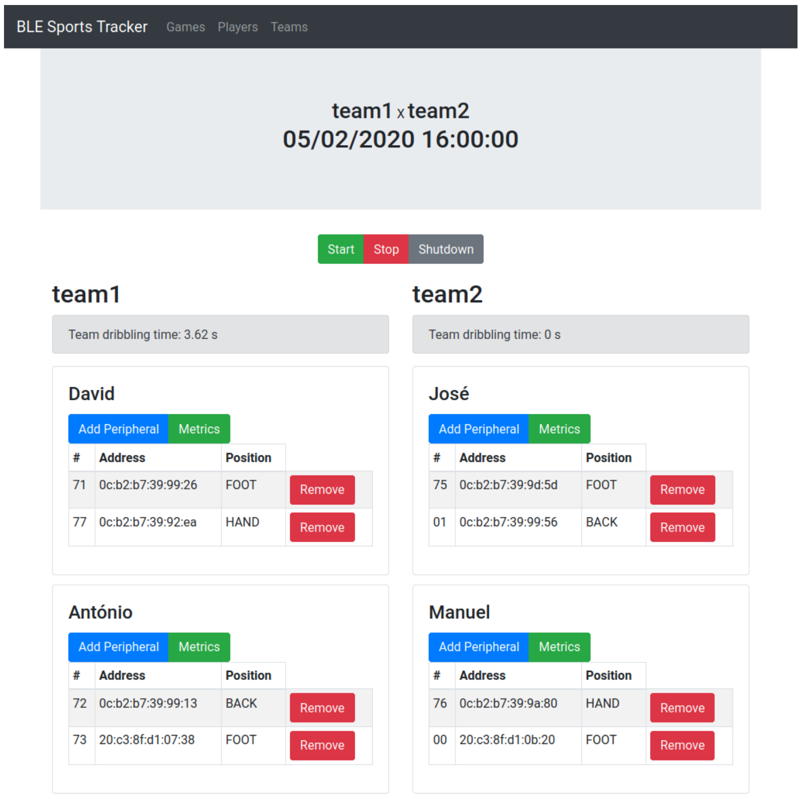
Select Date: 2020-02-17 10:35:19

team1 x team2
27/01/2020 11:30:00

team1 x team2
05/02/2020 16:00:00

(b) Add Game

Figure 4.9: Games List Page



team1 x team2
05/02/2020 16:00:00

team1

Team dribbling time: 3.62 s

David

#	Address	Position
71	0c:b2:b7:39:99:26	FOOT
77	0c:b2:b7:39:92:ea	HAND

António

#	Address	Position
72	0c:b2:b7:39:99:13	BACK
73	20:c3:8f:d1:07:38	FOOT

team2

Team dribbling time: 0 s

José

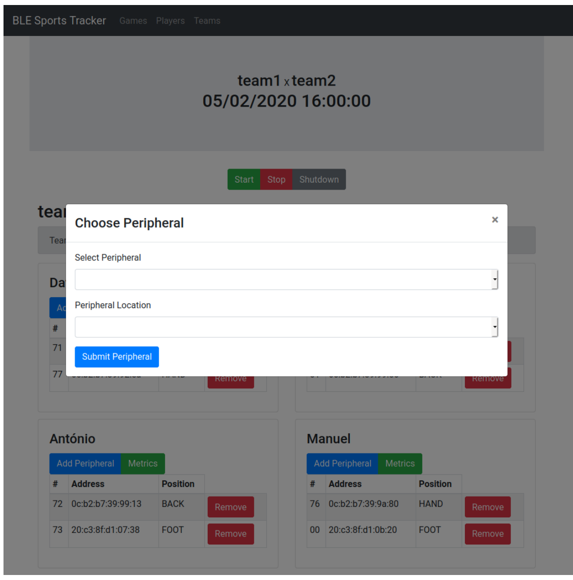
#	Address	Position
75	0c:b2:b7:39:9d:5d	FOOT
01	0c:b2:b7:39:99:56	BACK

Manuel

#	Address	Position
76	0c:b2:b7:39:9a:80	HAND
00	20:c3:8f:d1:0b:20	FOOT

Figure 4.10: Games Detail Page

of this game (start, stop and shutdown them), and can view the game metrics for each player, as shown in Figure 4.11b, and for the team. In this system implementation, the only team metric being calculated is the total time of dribbling of one team, from which we can infer what was the ball possession of each team during the game.


BLE Sports Tracker Games Players Teams

team1 x team2
05/02/2020 16:00:00

Start Stop Shutdown

Choose Peripheral

Select Peripheral

Peripheral Location

Submit Peripheral

António

Add Peripheral Metrics

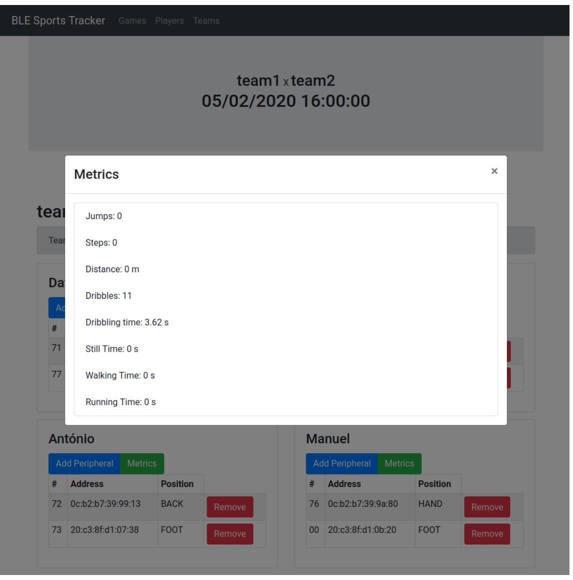
#	Address	Position	
72	0c:b2:b7:39:99:13	BACK	Remove
73	20:c3:8f:d1:07:38	FOOT	Remove

Manuel

Add Peripheral Metrics

#	Address	Position	
76	0c:b2:b7:39:9a:80	HAND	Remove
00	20:c3:8f:d1:0b:20	FOOT	Remove

(a) Assign Peripheral to Player


BLE Sports Tracker Games Players Teams

team1 x team2
05/02/2020 16:00:00

Metrics

Jumps: 0

Steps: 0

Distance: 0 m

Dribbles: 11

Dribbling time: 3.62 s

Still Time: 0 s

Walking Time: 0 s

Running Time: 0 s

António

Add Peripheral Metrics

#	Address	Position	
72	0c:b2:b7:39:99:13	BACK	Remove
73	20:c3:8f:d1:07:38	FOOT	Remove

Manuel

Add Peripheral Metrics

#	Address	Position	
76	0c:b2:b7:39:9a:80	HAND	Remove
00	20:c3:8f:d1:0b:20	FOOT	Remove

(b) Player's in-game metrics

Figure 4.11: Game Detail Operations

Chapter 5

Results

Table 5.1: Percentage of data loss (Raspberry Pi 3b+).

	0,6m	1,2m	3m	6m	9m	12m	15m	18m
1 Device	4,27	4,39	11,25	18,22	28,04	15,43	28,16	19,79
3 Devices	5,65	7,68	12,50	18,79	21,76	16,60	39,50	26,78
6 Devices	6,52	6,05	15,53	16,31	21,64	19,77	32,90	21,78
9 Devices	6,17	7,63	12,80	19,98	28,65	36,62	40,37	32,96

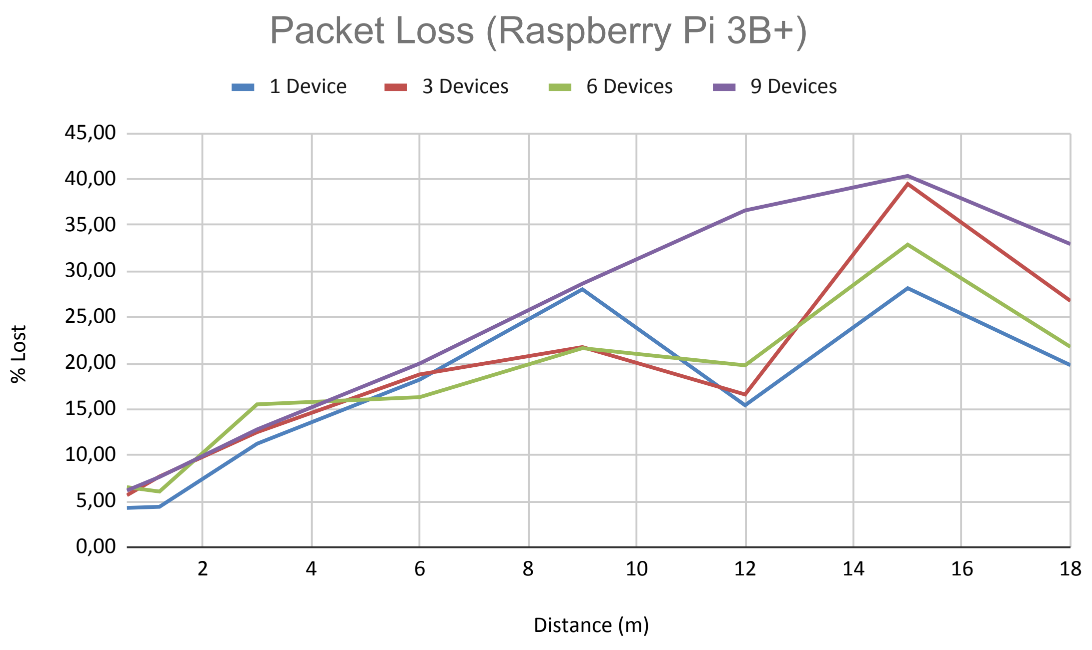


Figure 5.1: Percentage of data loss (Raspberry Pi 3b+)

1. Comunicação - Perda de pacotes IMU -> rPi
    a) raspberry Pi 3b+
    b) raspberry Pi 4
2. Fiabilidade dos algoritmos
    a) Dribbles
    b) Saltos - Os saltos têm de ter mais de 30cm, porque o algoritmo apenas deteta estes, se não confundia com corrida
    c) Passos/Distância
3. Apresentação dos dados

CHAPTER 6

Conclusions and Future Work

6.1 Conclusions

Revisitar os objetivos e comparar o trabalho feito com aquilo que foi definido. Avaliar o resultado final.

6.2 Future Work

Bibliography

[1] Dataconomy. *BIG DATA IS CHANGING THE FUTURE OF NBA SCOUTING*. 2017. URL: https://dataconomy.com/2017/08/big-data-changing-nba-scouting/ (visited on 07/11/2019).

[2] Kinexon. *The next era of sports analytics*. URL: https://kinexon-sports.com/solution/ (visited on 07/09/2019).

[3] KnowledgeBiz. *KnowledgeBiz*. 2019. URL: https://www.knowledgebiz.pt/ (visited on 07/11/2019).

[4] R. Socher. *NBA Loves KINEXON, It's Now the League's Most Used Wearable Tech*. 2018. URL: https://www.wearable-technologies.com/2018/09/nba-loves-kinexon-its-now-the-leagues-most-used-wearable-tech/ (visited on 07/11/2019).

[5] N. announces multiyear partnership with Sportradar and S. Spectrum. *NBA Communications*. 2016. URL: https://pr.nba.com/nba-announces-multiyear-partnership-sportradar-second-spectrum/ (visited on 07/11/2019).