Development and design of a virtual running pacesetter

Thato Matona
22127011

Report submitted in partial fulfilment of the requirements of the module Project (E) 448 for the degree Baccalaureus in Engineering in the Department of Electrical and Electronic Engineering at Stellenbosch University.

Supervisor: Dr J. A. A. Engelbrecht

June, 2022

Acknowledgements

I cannot begin to express my thanks to Dr Engelbrecht for his guidance and being my source of calm when I got confused. I will forever appreciate you for agreeing to walk down this path with me.

I would like to thank my brother, Thabang Matona, and grandmother, Christina Masiteng, for their support throughout my academic career. Thank you for your love, support, encouragement and belief in me. Your contribution to my successes is immeasurable.

I would like to thank my friends for their time. Thank you for listening to my endless ideas and helping me check my work, I appreciate you all.

I would like to thank my number 1 cheerleader, my late mother. Thank you for your everlasting love, support and presence in my life. Thank you for all the lessons you have taught me and the example you have led. You are a big portion of everything good that I embody.

Lastly, I would like to thank everyone else who helped me with my academics in the past few years and was not mentioned above. I am truly thankful to you for your contributions.

Stellenbosch UNIVERSITY IYUNIVESITHI UNIVERSITEIT

forward together
sonke siya phambili
saam vorentoe

Plagiaatverklaring / *Plagiarism Declaration*

1. Plagiaat is die oorneem en gebruik van die idees, materiaal en ander intellektuele eiendom van ander persone asof dit jou eie werk is.
   Plagiarism is the use of ideas, material and other intellectual property of another's work and to present is as my own.

2. Ek erken dat die pleeg van plagiaat 'n strafbare oortreding is aangesien dit 'n vorm van diefstal is.
   I agree that plagiarism is a punishable offence because it constitutes theft.

3. Ek verstaan ook dat direkte vertalings plagiaat is.
   I also understand that direct translations are plagiarism.

4. Dienooreenkomstig is alle aanhalings en bydraes vanuit enige bron (ingesluit die internet) volledig verwys (erken). Ek erken dat die woordelikse aanhaal van teks sonder aanhalingstekens (selfs al word die bron volledig erken) plagiaat is.
   Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism

5. Ek verklaar dat die werk in hierdie skryfstuk vervat, behalwe waar anders aangedui, my eie oorspronklike werk is en dat ek dit nie vantevore in die geheel of gedeeltelik ingehandig het vir bepunting in hierdie module/werkstuk of 'n ander module/werkstuk nie.
   I declare that the work contained in this assignment, except where otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

22127011 Studentenommer / *Student number*	 Handtekening / *Signature*
T.Matona Voorletters en van / *Initials and surname*	June, 2022 Datum / *Date*

Abstract

English

The running community has seen great improvements in the gear used to aid runners in their quests for being their best selves. However, current road running does not possess optimal pacing solutions. This project aims to develop and design an autonomous running pacing system for optimal running, moreso, for road and cross-country running. A way to source running route data is found. Then an autonomous pacer is developed. Then the best running technique is introduced and an algorithm to deploy the technique for the autonomous pacer is designed.

The location difference of a runner and the proposed location by the autonomous pacer must be visualised. The system will be tested by first measuring if the autonomous pacer completes a given route in a proposed time (assuming it's a realistic time). Secondly, by measuring if the autonomous pacer realistically adjusts the paces accordingly throughout the run with regard to elevation changes in accordance to the recommendation of the best running technique.

Route or course, stick on one...

Afrikaans

Why???

Contents

List of Figures

List of Tables

Nomenclature

Variables and functions

P	Runner's power (W)
P_s	Specific runner's power ($\mathrm{W\,kg^{-1}}$)
P_r	Running resistance (W)
P_a	Air resistance (W)
P_c	Climbing resistance (W)
m	Runner's mass (kg)
d_n	Distance between subsequent coordinates (m)
v	Runner's speed ($\mathrm{m\,s^{-1}}$)
C_r	ECOR ($\mathrm{J\,kg^{-1}\,m^{-1}}$)
ρ_{air}	Air density ($\mathrm{kg\,m^{-3}}$)
$A_r c_d$	Air resistance factor ($\mathrm{m^2}$)
v_w	Wind speed ($\mathrm{m\,s^{-1}}$)
c_d	Drag on the runner
A_r	Runner frontal area ($\mathrm{m^2}$)
i	Gradient of ascent or descent (%)
g	Gravitational acceleration ($\mathrm{kg\,m^{-2}}$)
R	Mean radius of the earth (m/km)
δ	Angular distance (radian)
ϕ	Latitude
ψ	Longitude
P_s	Specific Power ($\mathrm{W\,kg^{-1}}$)

Acronyms and abbreviations

APP	Application (CALL ALL OF THE INSTANCES APPLICATION)
LFR	Line-following Pacing Robot
ECOR	Energy Cost Of Running
GPS	Global Positioning System
4IR	Fourth Industrial Revolution
AR	Augmented Reality
VRP	Virtual Running Pacer

Chapter 1

Introduction

1.1. Background

In the past decade, the running community has seen a few leaps of improvement in the gear usage and studies conducted to aid runners in their quests for being their best selves. There has been an emergence of the running super shoes and data-based running technologies. Track running has been improved by the introduction of wave light technology [1] but no solution on par with it is used in road or cross country running. Geographic changes in a runner's immediate surroundings for non-track running provides a challenge that most current pacing technologies do not solve. Within the running community, terms like "bonking" or "hitting the wall" are all too familiar. They are an expression of when a runner surpasses their lactic acid threshold level and effectively runs out of energy as shown in Figure 1.1. The threshold pace is the running pace at which a runner's blood concentration of lactic acid begins to rise exponentially. "Hitting the wall" often happens when a runner plans to run just below their threshold pace throughout their run and encounters a steep or lengthy hill. The effort required to keep the same pace on a hill is higher than the effort required to keep that same pace on a flat or downhill. Most runners knows this but still, runners plan their runs for a constant pace when trying to beat their own personal records on uneven surfaces. Because it takes efforts close to the runner's lactate threshold to beat their record, runners surpass this threshold trying to keep the same pace on a course with a positive gradient. This results on a run not as optimal as intended. This project will focus on developing a virtual running pacesetter (VRP) to enable runner of all levels to run more optimally paced runs, specifically during road and cross country training and racing.

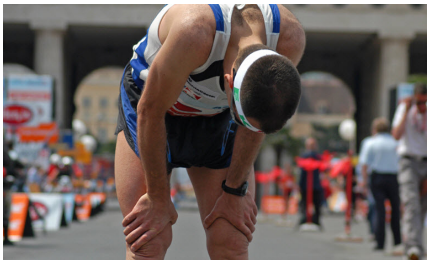


Figure 1.1: Hitting the wall

1.2. Socio-economic Implications

With the adoption of the fourth industrial revolution (4IR) technologies, it is crucial that a sustainability assessment is performed before any technology is built. This section will go over the economic and social implications of adopting technologies such as the VRP which this project aims to develop. The VRP of this project can be classified as automation. It automates and significantly improving a task which has been performed by human beings, mostly at elite running levels. Most major running events have human pacesetter. In elite races, these human pacesetter are enabled to make a living without necessarily having to worry about being amongst the very best athletes who collect prize money. Running is nothing if not a community sport and runners enjoy the human interaction that occurs with human pacesetters even during amateur events. A shift in mindset and breaking of the status quo is necessary for the success of technologies such as the VRP. The potential benefits of the VRP are used to justification for it's continued development. The benefits are:

- Promoting the use of technology to improve lives.
- Positive use of technology to improve athletic performances.
- Raises the competitiveness within the sport of running, improving the sport's reach.
- Encouraging unity and collaboration between competitors. Pacesetters and spectators act as a source of encouragement. Without the designated pacesetters, all runners must be encouraged to encourage each other during competitions. It is harder to run alone than it is with someone else.
- Enables new markets and research opportunities around the sport of running, in turn enabling runners to make more money from running. This includes the human pacesetters, new and older runners.

1.3. Problem Statement

A virtual running pacesetter system must be developed for road and cross-country running. The system must provide a virtual runner that a human runner can follow to complete a run in a desired time. The VRP system must plan the running pace for a given running route. The goal is to provide a VRP that a human runner can follow to complete a run in a desired time with the least amount of effort deviation throughout the run, allowing for maximised performance.

1.4. Objectives

Overall, a VRP must be developed and tested. The developed VRP must return the correct a location profile enabling a human runner to know where they must be located for a given time instance of a run. To ensure that the VRP system works as intended, project objectives are set and must be achieved. The project objectives are as follows:

- Develop a route map recorder that uses a GPS sensor to record the running route (latitude, longitude, and elevation) before the run.
- Develop a route pace planner that plans the pace for the virtual runner over the entire route, taking into account elevation changes, given the route map and the desired finish time.
- Develop a virtual runner for the human runner to follow. The virtual runner must execute in real time during the race on a mobile device, smartphone or smartwatch.
- Testing of the individual components of the system.
- Testing of the integrated system.

1.5. Summary

Evidence. What was achieved. Broad sense.

1.6. Scope

There are a number of factors that contribute to optimal running for a given race or training session. Some factors can be generalised and controlled but some are runner specific and cannot be controlled. The factors range from pacing to runner bio-mechanics and genetics. This project will focus specifically on the pacing with respect to changes in elevation. Pacing with respect to weather and traction from the running surface is considered but not heavily focused on. The focus is also on road and cross country running, which are typically 11 minutes (4 km for elite athletes) to 4 hours for average marathon runners.

1.7. Report Layout(Road-map)

Chapter 1: The project background, problem statement, objectives, summary and scope.
Chapter 2: Highlights of different technologies related to this project and discussion important information relating to the solution of the project problem.

Chapter 3: A system overview including hardware and software considerations.
Chapter 4: Detailed design including sourcing of test data and explanations of the algorithm used.
Chapter 5: A discussion and illustration of how the algorithm can be used practically.
Chapter 6: Experiments and results including a brief description of the design and setup of the experiments, the actual experiments, experiment results and a discussion of the results.
Chapter 7: The overall system summary, project shortcomings and recommendation for future work.
Appendices: The appendices include the project schedule, ECSA outcomes compliance, encountered problems and solutions, general notes, important results (will be changed to code dump).

Chapter 2

Literature Review

This chapter presents a literature review of background information, related work, and existing systems that are relevant to the development of a virtual pacesetter. It also draws some important conclusions from the presented literature.

The background information includes an overview of the different pacing strategies that are employed by runners, and the role of human pacesetters and their pacesetting techniques. The optimal running technique, which is based on the running model and the cori cycle, is also discussed. Finally, an overview of existing robotic and virtual pacesetter systems is provided.

2.1. Background Information

This section discusses important information required to understand the solution, its foundation, effectiveness and limitations.

2.1.1. Runner's Pacing Strategies

The article titled "How to pace your run" on the Expert Advice's website, [2], explains that runners employ a host of different running strategies. It suggests that runners should keep the same steady pace for road runs and vary their pace on trail runs even walking on some climbs instead of running. Its also suggests that it could be beneficial to start slower and incrementally speed up through the run.

2.1.2. Human Pacesetters and Pacesetting Techniques

A human pacesetter is a runner that is designated to run at a specific pace to encourage athletes to not run sections of the race too quickly or too slowly. Human pacesetters for track races aim to run at the same pace for the distance they are designated to run. This distance varies from pacesetter to pacesetter. It is based on the pacesetter's experience and fitness. For the marathon, understanding that not all parts of a marathon are the same in terms of effort is an important part of a human pacesetter's job. They are able

to guide runners more conservatively in the beginning of the race in order to ensure they have more energy and strength for later in the race when they need it. Good pacers will also guide their runners to run a slower pace up hills and then quicken their pace to gain that time back on any downhill stretches. One big mistake that new runners make is that they start out too fast. According to the article “The Marathon Pacer – Should you run with one?” [3], good pacesetters help runners hold back in the beginning of a race and then pick up the pace later.

2.1.3. Hitting The Wall

”Hitting the wall” or ”bonking” is a condition of sudden fatigue and loss of energy which is caused by the depletion of glycogen stores in the liver and muscles. The condition can usually be avoided by ensuring that glycogen levels are high when the exercise begins, maintaining glucose levels during exercise by eating or drinking carbohydrate-rich substances, or by reducing exercise intensity.

The body breaks down glucose to produce energy and lactate as a byproduct. As lactate is being produced, hydrogen ions are formed and lower the pH of the blood and thus making the muscles acidic. Through a process called the Cori Cycle [4], lactate is converted back to glucose - the source of energy production - removing the hydrogen ions. The cori cycle is illustrated on Figure 2.1. ”Hitting the wall” occurs when the build up of hydrogen

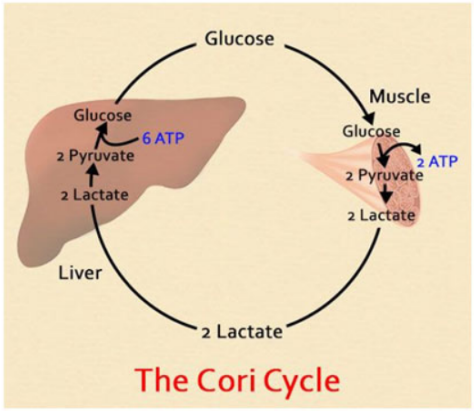


Figure 2.1: The Cori Cycle

ions, or proportionally lactic acid, occurs faster than the body can convert lactate back to glucose and clear up the hydrogen ions. Unfortunately, even running at a steady pace on a flat course will eventually lead to an excessive lactic acid build up as no runner possesses infinite amounts of glucose reserves. For this reason, it is wildely suggested [5] that runners take in carbohydrate and sugar rich energy gels or drinks during long runs. In so doing, it can be assumed that excess lactic acid above the threshold occurs not because of a lack of glucose in the body but because of the rate of the lactic acid build-up. This assumption is

made to better understand and quantify the effects of environmental change on the runner. An optimal running technique can aid in a runner in properly managing the rate of lactic acid build up in the runner's muscles.

2.1.4. The Running Model

Figure 2.2 shows the running model. The running model is a simplification of the power required from a runner to run at a certain speed.

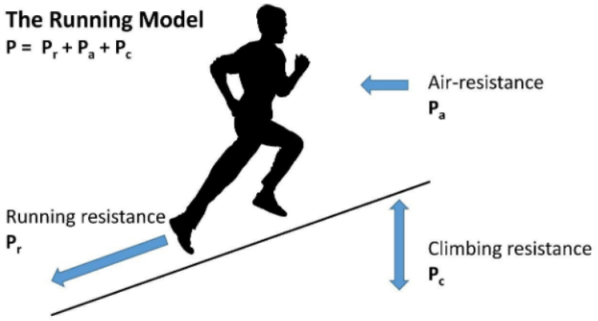


Figure 2.2: The running model

The running model is described in the book titled *The Secret Of Running* [6]. The book quantifies the required running effort as the runner's output power, P. The runner's power is the sum of the power of the running resistance P_r, the air resistance P_a and the climbing resistance P_c. P must be greater than zero for there to be running motion as will be shown.

The output power is proportional to the Energy Cost Of Running, ECOR. The energy cost of running is the energy cost per unit distance normalized to body mass. The unit of ECOR is joules per kilogram per metre. The energy cost of running is a runner-specific variable and is affected by many factors, including age, training, stride length, cadence, shoe weight, wind resistance, and air density (CITE, CHANGE DEFINITION). A focus on the power and ECOR is better than a focus on pace for optimal running. According to Stryd, [7], as the power increases, the runner must slow down and as the power decreases, the runner must speed up.

In physics, power is defined as the change in energy per unit time. The unit of power is Watt. In mechanical systems, power is defined as the product of the force acting on an object and the object's velocity. *The Secret Of Running* uses the above-mentioned definitions of power to describe the different powers as:

- The **running resistance** is a product of a runner's mass, speed and ECOR, C_r.

$$P_r = C_r * m * v \tag{2.1}$$

- The **air resistance** is the product of the aerodynamic force acting on the runner and the runner's speed. The aerodynamic force is a function of the air density ρ_{air}, the air resistance factor $c_d A_r$, and the square of the sum of the runner's speed v and the wind speed v_w.

$$P_a = \frac{1}{2} p_{air} c_d A_r (v + v_w)^2 v \tag{2.2}$$

- The **climbing resistance** is a product of the gradient, i, the runner's mass, ,, the gravitational acceleration, g, and the runner's speed, v.

$$P_c = imgv \tag{2.3}$$

  The gradient, i, translates the forward speed, v, into a vertical speed, iv, and the climbing resistance, P_c, therefore represents the power used to change the potential energy of the runner due to elevation. It is positive for a positive gradient, when running uphill, and negative for a negative gradient, when running downhill.

The runner's power is,

$$P = cmv + \frac{1}{2} p_{air} c_d A_r (v + v_w)^2 v + imgv \tag{2.4}$$

When the a runner is in motion, P must be greater than zero since every human being has mass and always expends energy when running, and their speed will not be zero when running. For a positive gradient, P increased and for a negative gradient, it is decreased.

2.1.5. The Optimal Running Technique

In athletics, a split is defined as the time it takes to complete a specific distance. The most commonly used splits are the mile and kilometer splits. In the past, an emphasis was placed on even splits - running at the same pace - for road and cross country running. This is stated in the conclusion of the article, "Describing and Understanding Pacing Strategies during Athletic Competition" [8]. According to the article, it is understood that running performance can be significantly influenced by the distribution of work during the run. Work is the transferring of an amount of energy. The article suggests that varying pace may be optimal under varying running conditions such as race duration, course geography and environmental conditions such as wind and temperature. It also states that for runs of a duration greater than 4 hours, evidence suggests that athletes may progressively reduce speed. An interesting admission in that article expresses doubt on which pacing technique represents an optimal running scenario and that further research is required. According to Stryd [7], maintaining a constant power is the most optimal running technique to ensure runners run perfectly paced runs.

2.2. Robotic and Virtual Pacesetters

This section provides an overview of existing robotic and virtual pacesetters.

2.2.1. PUMA BeatBot And Other Line-following Pacerbots

In 2016, Puma launched the Puma BeatBot [9]. It is shown on subfigure 2.3a.

(a) The Puma BeatBot **(b)** Another Line-following Pacerbot

Figure 2.3: Line-following robots

It is a line-following pacing robot (LFR) capable of guiding a runner along a track at constant pace depending on the time they want to beat. This type of technology pioneered automated running pacesetting and since then several replicas like the one on figure 2.3b have been made. LFRs are usually connected to a smart phone which is used to input the desired completion time and distance to be travelled. They calculate the constant average running pace. The robots require to be placed on a line, usually along a running track. They use cameras, ultrasonic sensors, actuators and motors to follow the line they are placed on while maintaining the calculated pace for the specified distance. The Puma BeatBot did this well. However, the biggest flaw of such a design is that the technology cannot be used where there is no line to follow. Another concern is the sophistication and cost of the hardware used to build such robots. The robots also do not provide the most optimal running for users with changes in the runner's surroundings. The technology is not robust. (INSPIRES IDEA AND NEED FOR IMPROVEMENT)

2.2.2. Ghost pacer

The Ghost pacer [10] is a pacing tool aimed at allowing runners to have a target to following when aiming for a certain running time. It uses augmented reality (AR) headsets to place an augmented runner besides the human runner. When instructed to start, the augmented runner runs at the desired running pace along a given route. Figure 2.4 shows the created augmented runner seen when using the AR headsets.

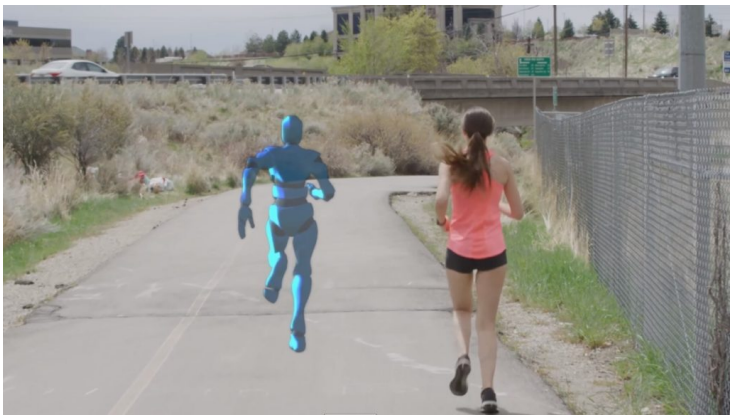


Figure 2.4: The augmented runner created by the ghost pacer

Like the LFRs discussed in the previous section, the Ghost pacer is not robust and is also made from sophisticated and costly hardware. This significantly reduces its accessibility. (INSPIRES NEED FOR FURTHER IMPROVEMENT)

2.2.3. Zombies, Run

Zombies, Run [11] is an interactive fitness application that engages runners into a post apocalyptic story and prompts them to walk jog or run. It prompts runners to change their speed depending on whether or not they are chased by zombies. Figure 2.5 illustrates how the application works.

Figure 2.5: Illustration of how Zombies, Run works

It proves the possibility of pace control for runners and illustrates the extent - how fun and interactive - pace controlling can be made. It contains a pace controlling element, which is what the autonomous pacer aims to do. However, the objectives of the pace control in Zombies, Run are significantly different from an automated pacesetter for serious running. Zombies, Run is aimed at making running fun while automated pacesetters are aimed at optimising running input vs output. (INSPIRES LIGHT/COLOUR INDICATORS)

2.2.4. Stryd

Stryd [7] is a footpod that analyzes the movement of a runner's foot and estimates their power output. It uses motion-capture sensors and wind capturing technology to understand how hard a runner is running. How hard a runner is running is measured as the power output of the runner. The footpod is mounted on a runner's shoe and connected to a phone or smartwatch. The footpod is shown on figure 2.6.

Figure 2.6: The Stryd footpod mounted on a runner's shoe

Stryd's usage of a power based metric and not a pace based one to optimise running more accurately represents the objectives of this project. However, Stryd has the disadvantage of requiring body mass calibration making it a runner specific product. The VRP will act as a ready to use tool and generalise the effect of changing environments for any runner anywhere in the world. (INSPIRES ECOR over PACE)

2.3. Conclusions From The Literature

The best pacing strategy for road running is running even splits. The best pacing strategy for trail running is to vary the pace as a function of the gradient. According to the running model, the power required from a runner is the sum of the running resistance, the air resistance, and the climbing resistance. Due to the Cori cycle, runners experience a sudden fatigue and loss of energy ("hit the wall") if their running power is too high for too long. The optimal running technique is to maintain a constant running power, which implies that a runner should decrease their pace when climbing, and increase their pace when descending. A virtual pacesetter for trail running should therefore adjust the pace as a function of the gradient.

Chapter 3

System Overview

This chapter introduces and explains the functions of the system's components and how each component interacts with other components. Figure 3.1 shows the system and its components.

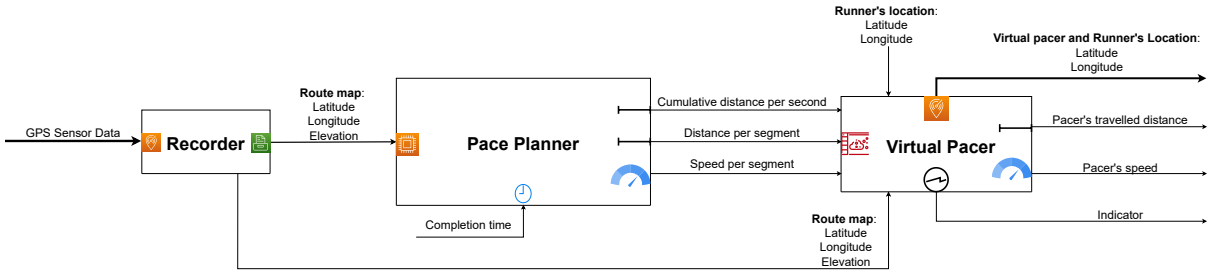


Figure 3.1: System diagram

Before the run begins, the recorder records route data and stores it for preprocessing. The recorded route data's resolution is increased and made more consistent and is then passed to the pace planner. The pace planner uses the processed route data and the desired completion time from the human runner to create different segments of the route and assign different target paces for each segment depending on the elevation change on that segment. During the run and in real time, the virtual pacer uses the runner's current location to find out the distance the runner has ran since the start of the run and compares it to the distance the pacesetter has planned for every second. The output of the virtual pacer is location of both the human and the virtual runner, the virtual runner's travelled distance, the virtual runner's speed and an indication of whether the runner is behind, ahead or on pace.

3.1. Recorder

3.1.1. Record route

The first project objective is to develop a route map recorder. The recorder uses a GPS sensor to pre-record the route and store it as latitude, longitude and elevation data points before the race begins.

3.1.2. Clean route data

There are two main issues with the use of a GPS sensor in the application of this project. The first is GPS measurement noise and the second is that the representation accuracy of the route by the recorded route data. The recorded route data is susceptible to GPS measurement noise. In addition to the noise, there is no guarantee that the person recording the route data will record an accurate representation of the route. In instances where they stand still for while recording the route, the route data would imply that runners needed to stop at that point which should never happen during a challenging run.

As the old adage would say, garbage in, garbage out. Instead of depending on the data to be accurately recorded, this data cleaning step is performed to equally space the GPS coordinates throughout the route. This ensures that the pace planner is passed reliable and the expected running route data. This step also ensures that movement from one GPS coordinate to the next is realistic. This step allows for a realistic path to be followed without the need of collecting too many way-points along the route. The recorder passes the newly calculated latitude, longitude and elevation sets to the pace planner.

3.2. Pace Planner

Before the start of the run the pace planner plans the paces for the virtual pacer over the entire route. It receives the preprocessed route data from the recorder and uses it to calculate the total distance of the route and the average speed required to finish the route given the completion time. It separates the route into segments of approximately equal distance and recalculates the speeds to be ran on each segment as a function of the changes in elevation on the respective segments. The pace planner then outputs the planned cumulative distance to be covered per second throughout the run, the distance per segment and speed per segment. The pace planner outputs are then passed to the virtual pacer.

3.3. Virtual Pacer

A way to visualise the planned run and compare it to a human runner's progress in real time is required. The virtual pacer is designed to help the human runner visualise how far off they are from where they should be during the run. During the run, it periodically outputs the human runner's location, the pacesetter's location, distance covered, current speed and an indication of whether the human runner is on pace, behind pace or ahead of the planned pace.

Chapter 4

Detailed Design

This chapter describes the detailed design of the individual components of the system. It investigates the influence of the hardware and software choices on the project final product, briefly discusses alternative ideas which were considered, justifies the choices and mentions mitigation measures taken to reduce potential pacer failure risks.

4.1. Recorder

The recorder is required to record and store the route (latitude, longitude and elevation) before the run and translate it into a a route map of consistently spaced waypoints that are used by the pace planner.

4.1.1. Record route

To record and store the route (latitude, longitude, elevation), a compact GPS module, a micro-controller and a data storage unit such as an SD card were the initial choices to perform the tasks. However, a mobile device, such as a smartphone or a smartwatch, is more naturally suited for the purpose of this project. This is because it has built-in capabilities to collect and store running data and because of the wide usage of mobile devices for fitness activities.

An android device is used to record and store the latitude, longitude and elevation points that represent a route. The points are recorded using Matlab Mobile and the android device's GPS sensor at 1 Hz. The sampling rate is the usual update rate for fitness application. The recorded points are stored on Matlab's online cloud.

4.1.2. Clean data

The pace planner and virtual pacer will rely on resolution of the data for consistency in the pacing updates. The clean data function improves the consistency by increasing the resolution of the recorded route data and eliminating points which indicate stops during the recording of the data. The route is not recorded at a higher sample rate because of

GPS sensor noise. Cleaning the recorded route data eliminates the need for the data to be recorded at very slow speeds or a high sample rate to try and get a high resolution route. This approach reduces the course data file size. This also allows for the recorder to walk or run the route at any pace - changing or not - and still be able to stop and continue while recording. However, there is no control for when the recorder goes back. This is because there is no expectation for a recorder or runner to do this. Running courses rarely make runners do a u-turn as faster runners would clash with the slower oncoming ones. The assumption is that if a u-turn is made, it is part of the running course. The distance of the course and the distance from coordinate to successive coordinate is to be calculated. According to Simon Kettle [12], the distance between two latitude and longitude sets can be calculated using the haversine function as follows:

The haversine function is defined as follows:

$$\text{haversine}(\theta) = \sin^2(\tfrac{\theta}{2}) \tag{4.1}$$

The distance d_n between two latitude and longitude coordinate sets (ϕ_1, ψ_1) and (ϕ_2, ψ_2) can be calculated using the haversine function as follows:

$$a = \sin^2(\tfrac{\phi_2 - \phi_1}{2}) + \cos\phi_1 \cos\phi_2 \sin^2(\tfrac{\psi_2 - \psi_1}{2}) \tag{4.2}$$

$$c = 2\arctan_2(\sqrt{a}, \sqrt{1-a}) \tag{4.3}$$

$$d_n = Rc \tag{4.4}$$

where R is the mean radius of the earth, which is 6 731 km.

The resolution of the course data can be increased by decreasing the distance between subsequent coordinates. The resolution is controlled and is chosen as 1 m to balance simulation time and accuracy. The change in resolution was done by finding the coordinates at the start of the route and incrementing the position by the chosen resolution value with regard to the current bearing. This approach finds the coordinates 1 m away in the direction specified by the current bearing. This is done until the total incremental distance from starting coordinates is more than the previously calculated distance between the starting coordinates and the subsequent recorded coordinates. The process is repeated, starting from the last set of calculated coordinates to the next set of recorded coordinates with regard to the change in bearing. The new bearing, b_r, from last calculated coordinate (ϕ_c, ψ_c) to the next recorded coordinate (ϕ_r, ψ_r) is,

$$b_r = \arctan_2(\sin(\psi_r - \psi_c)\cos\phi_r, \cos\phi_c \sin\phi_r) - \sin\phi_c \cos\phi_r \cos(\sin(\psi_r - \psi_c) \tag{4.5}$$

Subfigure 4.1a shows new equally spaced coordinate sets represented by the red dots.

These sets are found from interpolating between the recorded coordinate sets represented by the black dots. When the bearing changes, a small angle is created as the incremental distance is not calculated from point 2. The small angle approximation, $cosx \approx 1$, is applied to show that the distance travelled by the pacer and the person recording the data is almost the same. The difference is negligible. Angle x is between point 2 and point 3's red and green lines. Equations from the Movable Type Scripts website [13] were used to find the coordinates at the incremental distance, 1 m. These equations are shown in 4.6 and 4.7 with δ being the travelled angular distance and θ the bearing, clockwise from true north. Equation 4.6 finds the next latitude point and equation 4.7 finds the next longitude point.

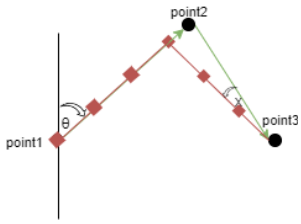
$$\phi_2 = \arcsin(\sin\phi_1 \cos\delta + \cos\phi_1 \sin\delta \cos\theta) \tag{4.6}$$

$$\psi_2 = \psi_1 + \arctan_2(\sin\theta \sin\delta \cos\phi_1, \cos\delta - \sin\phi_1 \sin\phi_2) \tag{4.7}$$

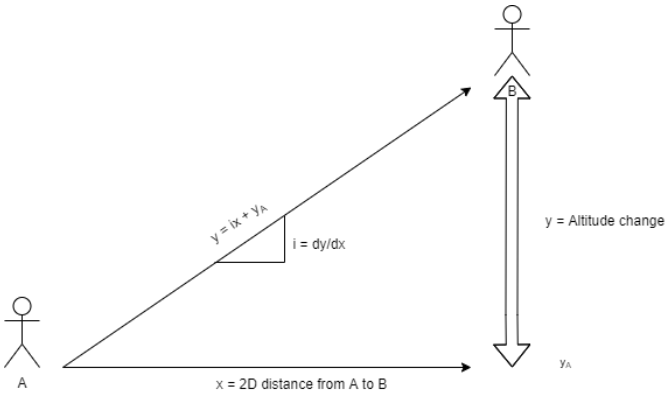
The resolution of the altitude data was also increased to match the changes in position. This was done by linearising the altitude change as position changed. The gradient, i, of the linear change becomes the altitude change between the subsequent recorded coordinate sets divided by the distance between the points. This is shown on equations 4.8 and 4.9 and on subfigure 4.1b.

$$i = \frac{dy}{dx} = \frac{y_B - y_A}{x_B - x_A} \tag{4.8}$$

$$y = ix + y_0 \tag{4.9}$$



(a) Interpolating new coordinate sets from the recorded sets



(b) Linearising altitude change for subsequent coordinate sets

Figure 4.1: Interpolating new coordinates and their respective approximate altitudes

4.2. Pace Planner

4.2.1. Distance and speed calculator

Using equations 4.2, 4.3 and 4.4, the total route distance, d_{total}, is calculated. With N as the number of coordinates representing the route and d_{n} being the distance between two consecutive waypoints, the total route distance is,

$$d_{total} = \sum_{n=1}^{N} d_n \tag{4.10}$$

Once the user inputs the desired completion time, $t_{\text{completion}}$, the required average speed is calculated as follows:

$$v_{average} = \frac{d_{total}}{t_{completion}} \tag{4.11}$$

The pace planner's maximum allowable pace is taken as 3 minutes 43 seconds per mile. This is the world record pace for the mile set by Hicham El Guerrouj. This translates to 7.14 $\mathrm{m\,s^{-1}}$. Average runners are expected to run much slower than this.

4.2.2. Segments and pace per segment

The route is divided into segments of equal distance. The division is done by a value for frequency of pace change, f_{pc}, and grouping the segments to have the same number of coordinates. For the case when the number of coordinates is divisible by f_{pc}, each segment will contain f_{pc} number of coordinates and all segments would be of equal distances if all subsequent coordinates are equidistant from each other. However, that is not always the case. An example to illustrate how a route is segmented if the number of coordinates is not divisible by f_{pc} is shown on Figure 4.2.

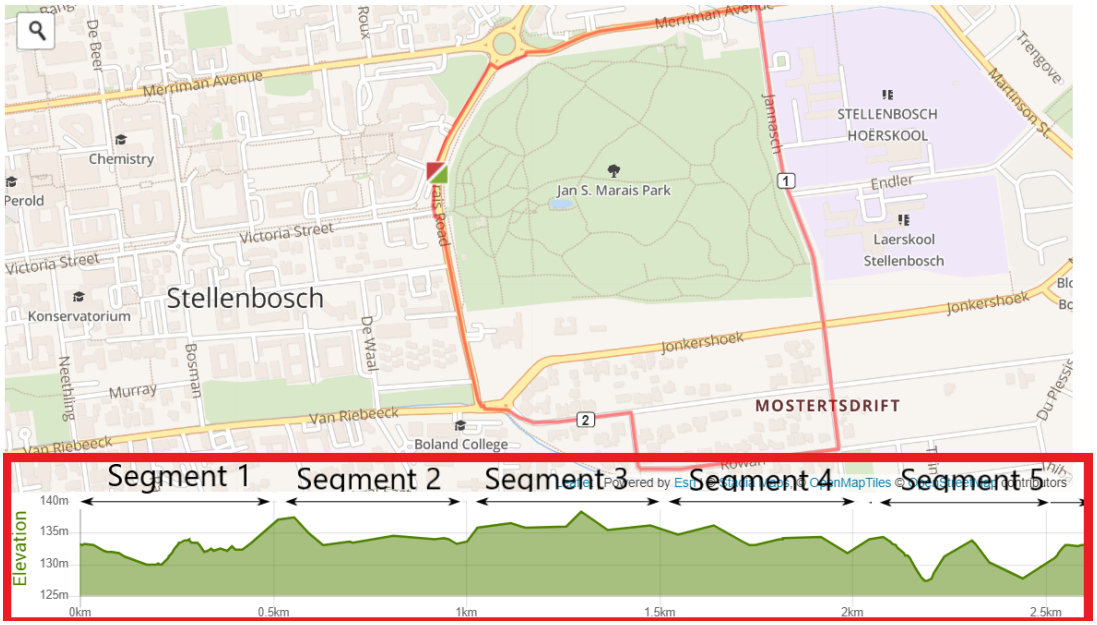


Figure 4.2: An example of how a running route is segmented

The distance per segment is depended on the chosen f_{pc}. A f_{pc} value of 15 coordinates is chosen to ensure that as much altitude change is represented while allowing a runner time to adjust to the new pace on each segment considering that the paces are below 7 $\text{m}\,\text{s}^{-1}$. This prevents the VRP from providing the runner with rapidly changing paces that are impossible to follow for any realistic desired completion time. For a f_{pc} value of 15 coordinates, the distance per segment will be smaller than the 500 m shown on Figure 4.2. The last segment is of shorter distance.

If there is a way to keep track of the energy lost during the run, then energy supplements can be taken to replace the lost energy. Individual metabolism factors are not considered as stated in the scope 1.6. A way to keep track of the energy lost dring a run is by controlling the energy loss during that run. Using the data from the distance and average speed calculator, the altitude changes are considered and the average speed on each segment is adjusted with respect to the elevation gain or loss on that segment. Then the distances to be travelled by the runner at every time interval is recorded and passed to the virtual pacer function block.

As stated in the scope in section 1.6, the effects of weather will not be focused on. By neglecting drag and assuming wind speed to be zero, equation 2.4 simplifies to,

$$P = C_r m v + i m g v \tag{4.12}$$

To eliminate the dependence on a runner's mass, the specific power,

$$P_s = C_r v + i g v, \tag{4.13}$$

is used instead of power. The ECOR is directly proportional to the specific running resistance, $P_{sr} = C_r v$. And also, the gradient is directly proportional to the specific climbing resistance, $P_{sc} = igv$. Consider a runner on a flat course. This means the gradient is zero, therefore,

$$P_s = P_{sr} = C_r v \tag{4.14}$$

When a gradient i is introduced, then the running speed v must be adjusted to keep the specific running power P_s the same. To calculate the adjusted running speed $v_{gradient}$ on the gradient, we equate the specific running power on a flat course $P_{s,flat}$ with the specific running power on a gradient $P_{s,gradient}$, as follows

$$P_{s,flat} = P_{s,gradient} \tag{4.15}$$

$$C_r v_{flat} = C_r v_{gradient} + igv_{gradient} \tag{4.16}$$

Note that we assume that the energy cost of running stays constant. We then solve for the adjusted running speed $v_{gradient}$ on the gradient

$$v_{gradient} = \frac{v_{flat}}{1 + \frac{ig}{C_r}} \tag{4.17}$$

Gravitational acceleration is constant, so if the ECOR is to be kept constant, the only factor that can affect the change in speed is the change in gradient ascent or descent. When the gradient on a segment increases, the new speed will be less than the old speed vice versa. The energy and specific power is keep constant resulting optimal running. With K as the number of segments, d_{k} as the distance per segment and v_{k} as the speed on the current segment, the new average speed for the whole course, $v_{\mathrm{newaverage}}$ is calculated as shown in 4.18.

$$v_{newaverage} = \frac{d_{total}}{\sum_{k=1}^{K} \frac{d_k}{v_k}} \tag{4.18}$$

The last segment is usually significantly shorter than the others, $d_{\mathrm{k}} >> d_{\mathrm{K}}$. Although d_{k}, with $k \neq K$, is relatively constant, v_{k} varies. Consequently, the time spent on each segment varies. The new speed per segment, v_{k}, is scaled such that the completion time is as specified by the user as shown on equation 4.19. This ensures that the overall average speed remains what it is calculated for on a flat course, therefore the finishing time remains what was specified.

$$v_k = v_k \times \frac{v_{average}}{v_{newaverage}} \tag{4.19}$$

4.3. Virtual Pacer

Matlab's geoplayer function is chosen as a way to visualise the results of the pace planner. It was chosen because it is capable of displaying real life map data from the use of only latitude and longitude sets. Avoiding the usage of programs or function blocks that required the full set of a gpx file to run a visualising simulation of map data was the main goal. A gpx format type is the typical map data file type. Geoplayer allows for isolation of only the required data for the autonomous pacer algorithm to use. This decreases simulation time and allows for more accuracy and resolution in the results.

The runner's location data is fed to the visualiser. The runner's distance travelled is compared with the pacer's planned distance to travel periodically. A sampling time of one second was chosen as is the typical sampling time for most fitness apps. Figure 4.3 shows an example of the output of the virtual pacer on matlab.

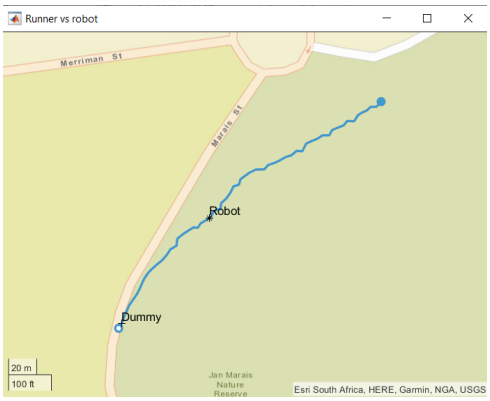


Figure 4.3: Virtual pacer example output

Equations 4.2, 4.3 and 4.4 are used to find the distance travelled by both the runner and pacer every second. As stated in 4.2.2, the pace on each segment remains constant. The pace of the VRP remains the same until the distance on the current segment has been ran. When the runner's distance travelled subtract the pacer's planned distance is less than pacer's current pace, then the runner is seen as being behind pace. In this case, a red light is shown. To indicate that the runner is within range of the pacer, the absolute value of the runner's distance travelled subtract the pacer's planned distance is smaller or equal to the current pace, the runner is seen as being on pace. In this case, a yellow light is shown. When the runner's distance travelled subtract the pacer's planned distance is more than pacer's current pace, then the runner is seen as being ahead of pace. In this case, a green light is shown. The runner must try to keep themselves within pace. If they are behind pace, they might be exerting less energy but will require too much energy to catch up if they still intend on finishing the run at the desired completion time. If the runner is ahead of pace, they are

most likely overexerting themselves and will not be running optimally. When the VRP is complete, the message, "Run must have been complete already" is displayed. Throughout the whole run, the runner is made aware of their and the VRP's position. If they follow the indicators, they will finish the run in the intended optimal running fashion.

The VRP does not have control for runner deviating from the running course and hence cheating. Because only the distance travelled is compared, if a runner deviates significantly from the route, the virtual pacer will show the deviation but the indicators will still run as if the runner is on the running course.

Chapter 5

Practical Implementation

This chapter describes the practical implementation of the VRP in hardware and software.

5.1. System Overview

Figure 5.1 below illustrates how the developed pacer could be implemented as software within a host device.

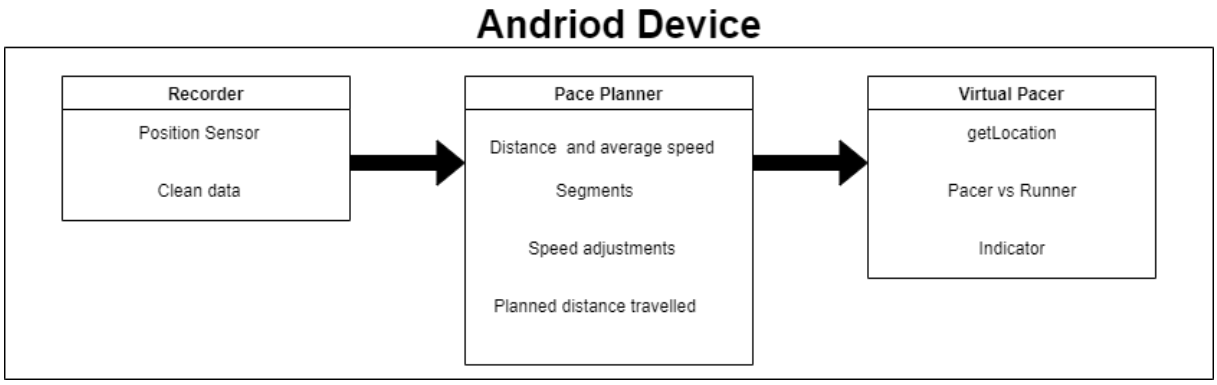


Figure 5.1: System overview on a host device

Route data is collected through the runner's android device and saved onto the host application. The VRP uses the route data and sets the location which the runner should be at given time intervals during their run. During the run, the runner's current location is compared with the location set by the VRP and a visual signal to the runner is given to indicate their progress relative to the pacer. This system can be used for any android device, including and not limited to phones and smart watches. This allows the solution to be accessible to many runners.

5.1.1. Recorder

The challenge here is acquiring usable running route data. To demonstrate usability of the VRP, route data is acquired through Matlab mobile and its sensor options. Potential host applications for the VRP's algorithm are popular fitness applications such as Strava. For this reason, an assumption that the host application is capable of collecting the data

required by the algorithm is made. This subsection describes the steps taken to collect route data through Matlab mobile and import it to Matlab as the input to the VRP's algorithm. It also describes how the theory to clean the recorded data is implemented in software.

Step 1

Firstly, Matlab mobile is downloaded from google play store onto the host device. The application is shown in Figure 5.2.

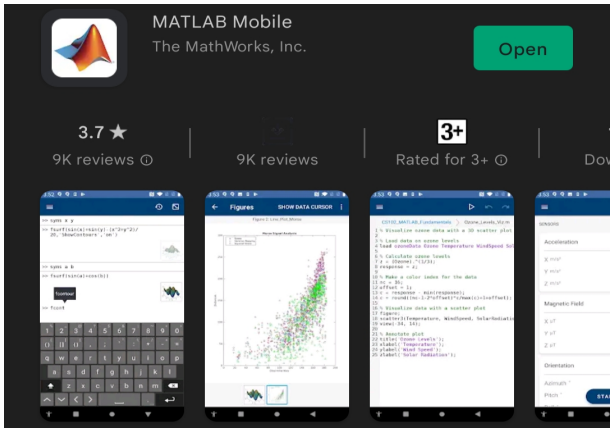


Figure 5.2: Downloading Matlab Mobile

Step 2

Once Matlab mobile is downloaded and installed onto the host device, a mobiledev object [14] is created to read sensor data from the host device. This is done on the command prompt of Matlab mobile as shown in Figure 5.3. If data is to be streamed to Matlab, the "logging" variable of the created mobiledev object must be set to 1.

```
>> m = mobiledev
```

Figure 5.3: Creating a mobiledev object

Step 3

Once step 2 is competed, the sensors of the host device can be accessed as shown in Figure 5.4.

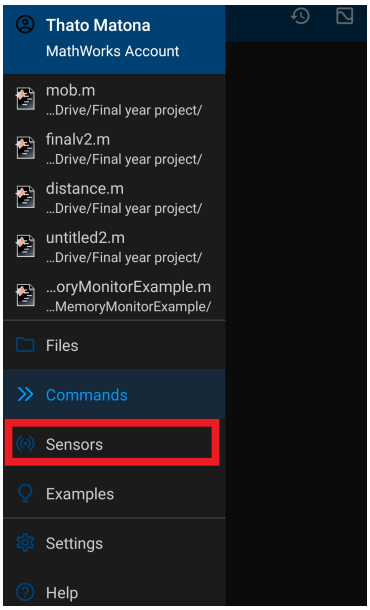


Figure 5.4: Accessing sensor functions

Step 4

For this application, the sensor is set to stream the recorded data to the same Matlab workspace Matlab mobile was working on at 1 Hz as shown in Figure 5.5. The sensors can also be set to log and save the recorded data onto the host device and the data can be manually imported onto the Matlab workspace.

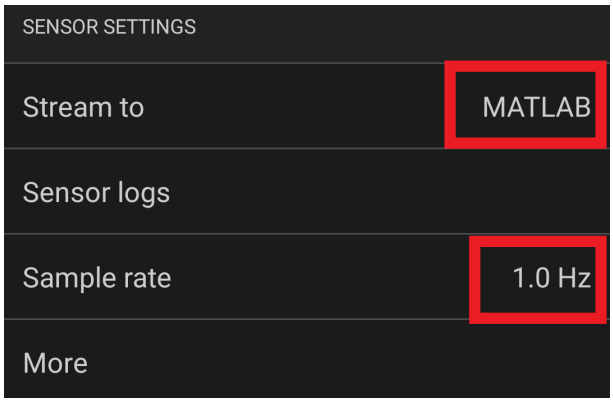


Figure 5.5: Setting up the streaming environment and sample rate

Step 5

When the setup is complete, the person collecting the data must go to the starting point of the intended running course and traverse the whole course. Position data shown in Figure 5.6 is recorded every second as specified by the sample rate in step 4.

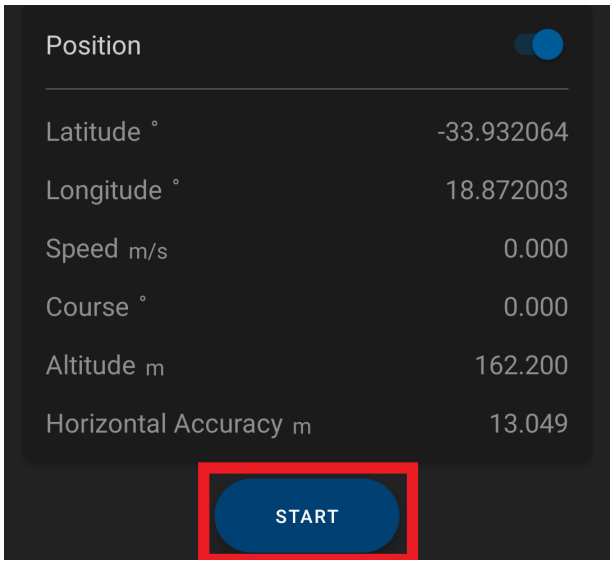


Figure 5.6: Recording the route data

Step 6

When the finish point of the course is reached, streaming must be stopped as shown in Figure 5.7, and the data saved onto Matlab.

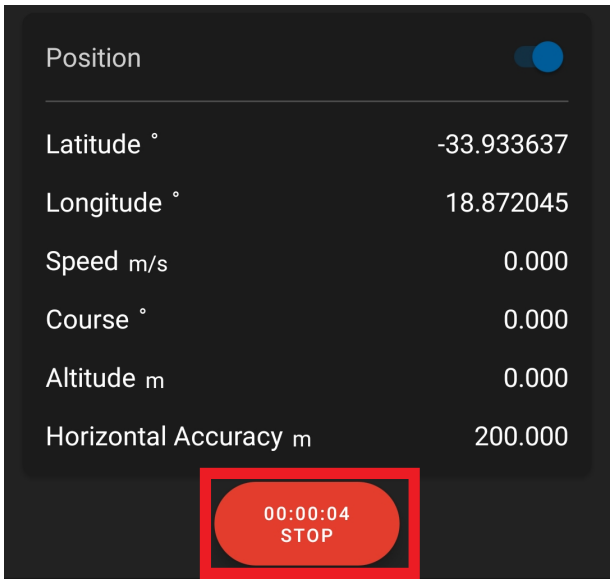


Figure 5.7: Saving the recorded route data

Once the above steps are completed, the recorded data can be preprocessed. Using equations 4.2, 4.3, 4.4 and 4.10, the distance between successive recorded coordinate points and the total course distance is found. This is shown in figure E.1. Figure E.2 shows how the route data is made more consistent. The set resolution is shown to be 1 m. The resulting distances between subsequent coordinate sets is shown in Figure E.4.

5.1.2. Pace Planner

The theory in section 4.2 is applied to create an algorithm that plans the paces for the VRP. The paces are translated to distance travelled. Figure 5.8 shows the functions of the pace planner.

```
[df, distance_per_segment] = distanceIncr3(oldlat,oldlong,size,fpc);
avgspeed = df(size)/x;
speed_per_segment = segmentTimes(avgspeed, alt, fpc, distance_per_segment,df(size));
dist = diste2(x,speed_per_segment,distance_per_segment,df,size);
```

Figure 5.8: Pace planner functions

The total course distance and distance per segment is calculated. This is shown in Figure E.5. Then the average speed is computed. Figure E.6 shows how the route is segmented and the gradient on each segment is found. Then also how the new speed on each segment is calculated. Once the distance and speed per segment is calculated, the distance per second is computed. This allows the pacer to know how much distance should be covered every second during the run. The distance per second varies depending on the segment. This process is shown on Figure E.8.

5.1.3. Virtual Pacer

The pacer is tested using a standard android device. The device used is the Xiaomi Redmi Note 8. The implementation allows for most standard android devices including smartwatches to be used.

It is important to note that the Robot.txt file shown in figure 5.10 would be produced by the host application. Figure 5.9 shows how the VRP output is produced. For the purpose of this project, the VRP output - Robot.txt - is imported from Matlab into the host device's memory as shown in Figure 5.10.

The android application was written specifically to display the position of the virtual runner such that it can be followed by a human runner. Once the run begins, the application collects the position updates of human runner every second in real time and store it on the mobile device's memory. The data is then used on Matlab to analyse the effectiveness of the virtual pacer.

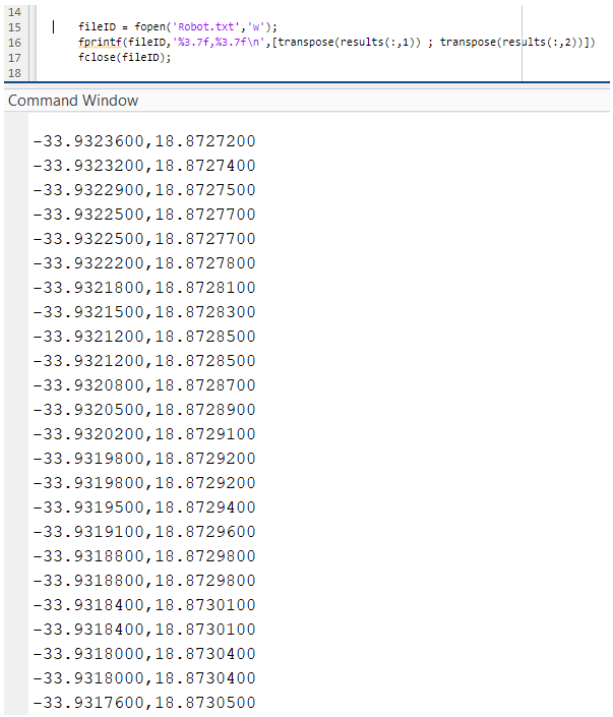


Figure 5.9: Pacer algorithm output

The planned location profile is reported as text file.

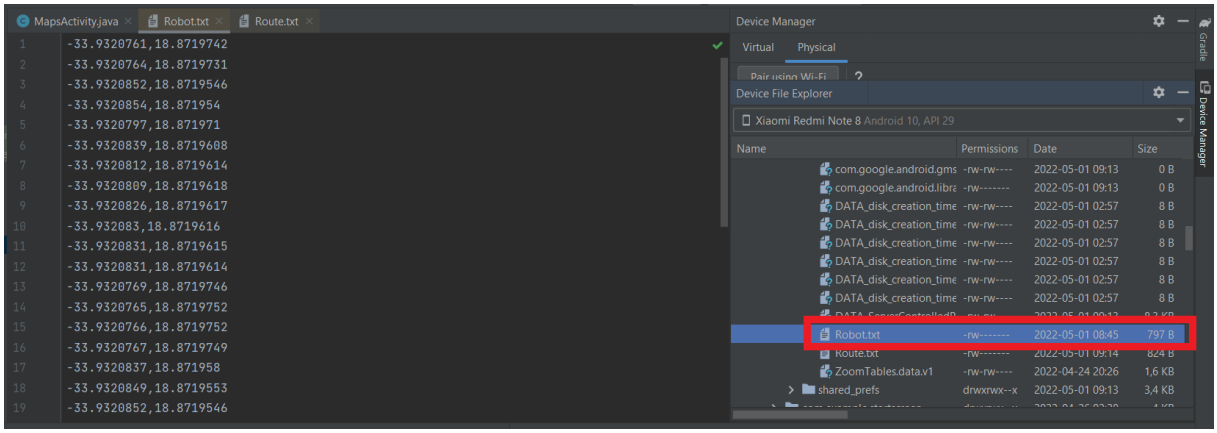


Figure 5.10: Importing the pacer algorithm's output into host device

When the application on the host device is started, the live location of the runner is updated every second and compared to the output of the pacer. The indicator logic is the same as mentioned in section 4.3. When the runner is within range of the pacer, a visual indication that they are on pace is given. Importantly, he runner will be aware of their relative position to the virtual pacer at all times. The virtual pacer compares the distance travelled by the runner and the distance by the VRP every second and gives indicators. The algorithm is shown on Figure E.10. The distance travelled is translated to a location profile which can be visualised on a map in real time on the host application. Figure 5.11 shows a real time execution of the application. The virtual pacer progressive position is

shown by the red location markers and the human runner's live location is shown by the blue google maps location button.

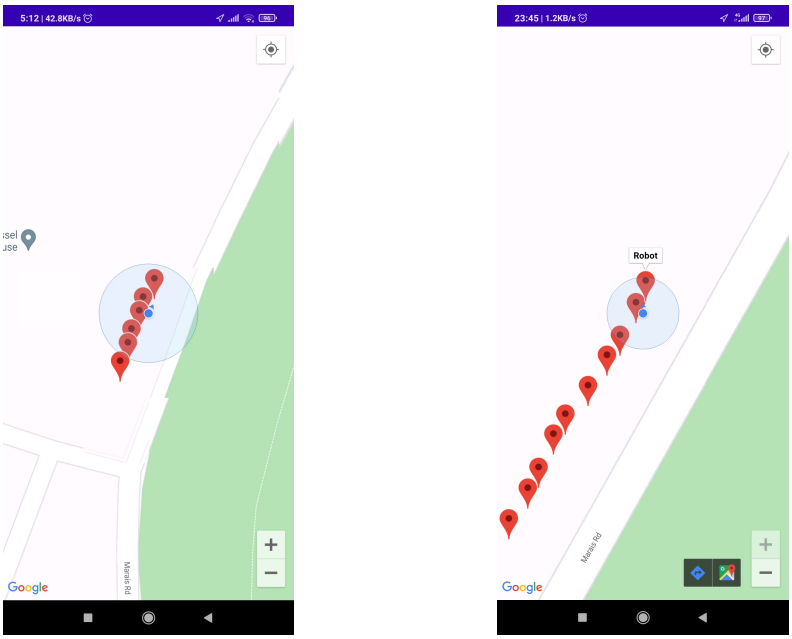


Figure 5.11: Screenshots of a real time execution of the application

5.2. Product Transition

For the purpose of this report, a standalone android application was created to illustrate how the VRP could be implemented for usage. However, the VRP can be integrated with existing fitness applications such as Strava, Zepp Life, NRC, Garmin Connect, Zombies Run and many more. The VRP was not integrated with matlab's android support package to maximize accessibility. As stated on the android studio IDE and on the android police website [15], the matlab android support package compiles for minimum API 30 android devices. Only 24.3% of android devices have an API 30 SDK or above. Runners are already using the popular fitness applications so it is sensible to improve their running experience and not change it by using matlab mobile for the VRP usage. Also, all of the mentioned fitness applications already collect the data required by the VRP to work. This would make the integration process close to seamless. The VRP's matlab code can be rewritten as java or kotlin code of the same functionality. The application can be developed for IOS devices too to increase accessibility. Like Zombie Run, an emphasis on the indicators can be made to engage users and improve interaction with the application. On the results, it will be shown how the visual cues can be made colours for every update. This can be translated to notification LED colours, screen colour pixel changes or even sound effects.

Chapter 6

Experiments and Results

This chapter discusses the target metrics, the experiment design procedure to test those target metrics. The results of the experiments are then presented and discussed and conclusions are made in relation to the project objectives.

6.1. Target Metrics

The pacesetter's **completion time** is the first metric of interest. The pacesetter is supposed to enable a runner to finish the given running route at a set time. The completion time of the pacesetter must match the desired completion time of the runner. The **distance travelled** by the runner and pacer is compared for every time instance and a **response** to the deviation from the pacer will be observed. The pacer is supposed to indicate if the runner is on pace or not. The **speed** with respect to **elevation** change is also measured to observe how the pacer adjusts the proposed speeds for changes altitude.

6.2. Experiment Design Procedure

In section 1.4, project objectives (develop and test a VRP system) are stated and will be tested in this chapter using the target metrics stated in section 6.1. The VRP must be able to adjust paces with respect to the elevation throughout the run and provide a location profile for the human runner to follow. This section tests the effectiveness of the VRP. This will be done by comparing the virtual pacer's completion time with the desired completion time. Then checking the speeds at different segments as a function of the gradient on those segments. Then also by getting the output location profile of the virtual pacer to see if it is attainable.

6.3. Experiment Results

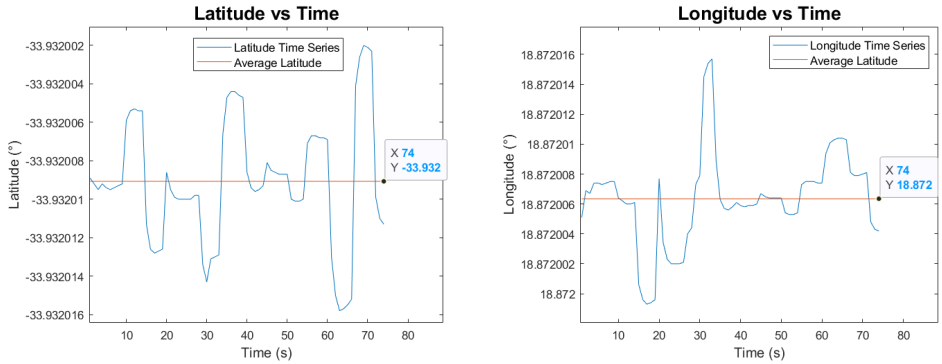
6.3.1. Recorder

The following tests were undertaken to test the GPS:

Static test

The static test is performed to determine the position measurement noise of the GPS sensor, and to evaluate whether it is small enough for the application of the VRP. The test is done by standing in one position and recording what the GPS measures as the latitude and longitude over time. The test was done for 74 seconds.

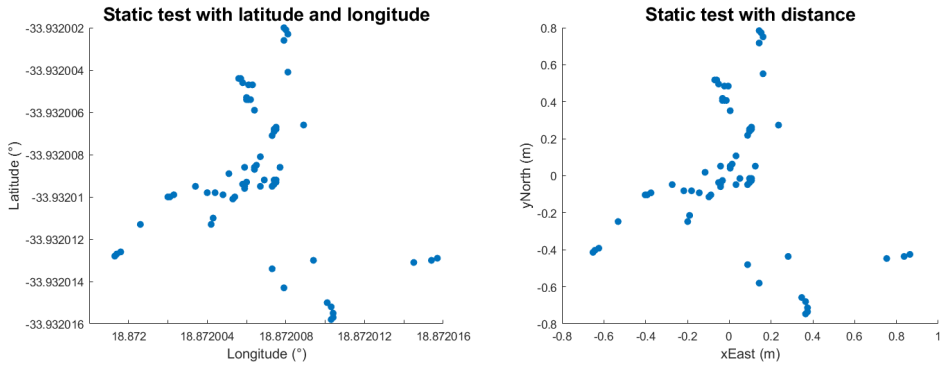
Figure 6.1 shows the latitude and longitude change over time.



(a) Latitude vs Time **(b)** Longitude vs Time

Figure 6.1: Static test latitude and longitude change over time

Figure 6.2 shows the positional change in the results of this test. Changes in latitude and longitude are shown in subfigure 6.2a. The average of all the latitude and longitude points was calculated and represents the origin used to calculate the changes in north and east as shown in subfigure 6.2b.



(a) Static test latitude and longitude plot **(b)** Static test positional change in metres

Figure 6.2: Static test positional change

The recorded latitude and longitude changes even when the points were recorded at the same point. The amplitude from the average of the measurements indicates sensor noise. In subfigure 6.2b the noise is shown in metres. The maximum error in the north is 0.8 m and 0.9 m in the east. The users of the VRP are expected to be able to run a marathon in under 4 hours. This translates to a pace of 2.92 $\mathrm{m\,s^{-1}}$. The measurement error is not significant enough such that a human runner running at 4 hour marathon pace would not

be able to quickly readjust after a few seconds if an error in their position or the virtual runner is made. The position measurement noise of the GPS will not significantly affect the applications of the VRP.

Altitude test

The altitude test is performed to determine the altitude measurement noise of the GPS sensor, and to evaluate whether it is small enough for the application of the VRP. The test is done by standing at the same altitude level over time. The test was done for 24 seconds. Figure 6.3 shows the results this test.

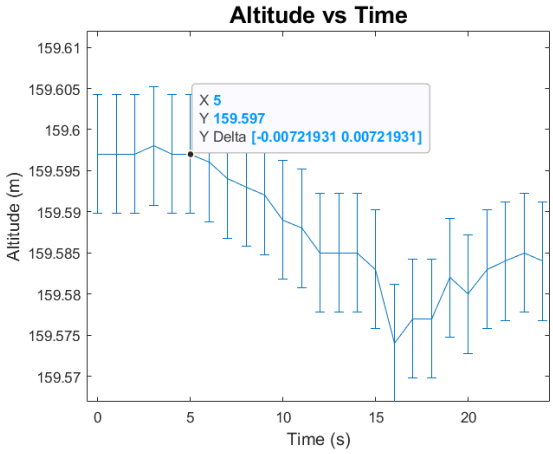


Figure 6.3: Altitude accuracy test results

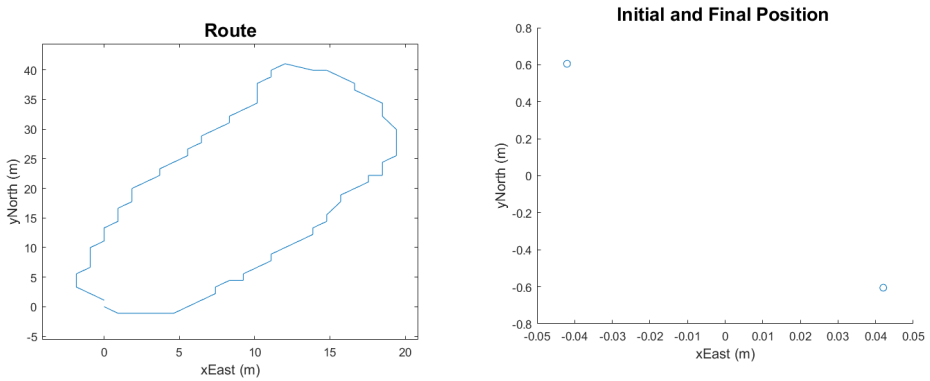
With the altitude at time t, a_t, the average altitude, u, and the total time, T, the standard deviation σ was calculated as follows:

$$\sigma = \sqrt{\frac{\sum_{t=0}^{T}(a_t - u)^2}{T}} \tag{6.1}$$

The standard deviation of the altitude measurements is 0.00721931 m. Considering the use of the altitude data on the project, as seen in the previous sections, the deviation in the measured altitude level is small enough to be negligible.

Repeatability test

The repeatability test is performed to determine the likelihood of getting the correct latitude and longitude position measurement from the GPS sensor. This test is done by standing in one position for 10 seconds then moving around and back to the initial position and then standing there for a further 10 seconds. The average latitude and longitude values are calculated for the first and last 10 seconds. These are the points when the recorder was static and the difference between these points represents the repeatability of the GPS measurements.



(a) Recorded route in metres

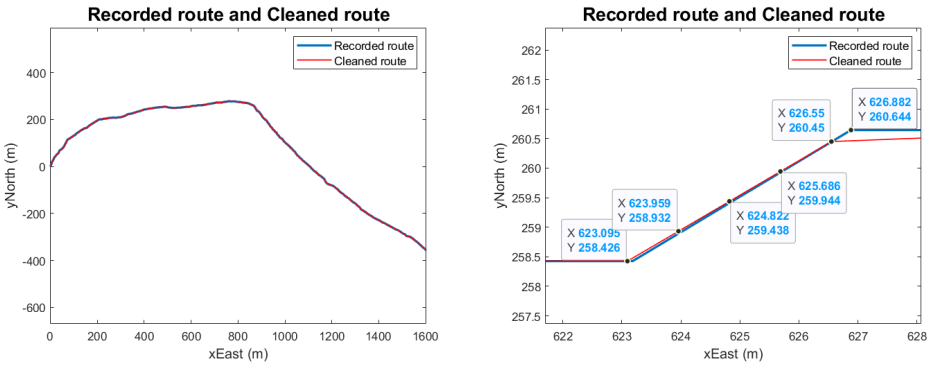
(b) Initial and final recorded latitude and longitude

Figure 6.4: Repeatability test results

Subfigure 6.4a shows the route in metres with the average of the first 10 recorded positions as the origin. Subfigure 6.4b shows the average of the first 10 recorded positions and the average of the last 10 recorded positions. The difference in the initial and final position is $1.0909e^{-}5$ in latitude and $9.0909e^{-}7$ in longitude. The difference in distance between the points on subfigure 6.4b is 1.46 m. This is less than the 2.9 m requirement of the VRP. The GPS sensor is capable of producing the same latitude and longitude positions for the same route. This means the VRP will receive reliable route data from the GPS sensor.

Clean data

The consistency of the recorded route data is to be improved. This is done by setting the resolution of the route data and creating a representation of the route with approximately equally spaced waypoints as explained subsubsection 4.1.2. A route of approximately 2 km was recorded. The route's resolution was set to 1 m. Subfigure 6.5a shows recorded route map in blue. The route map represents the route as was ran. The runner ran on the side walks and not entirely on the road when recording the route. The cleaned route is shown in red on subfigure 6.5a. Subfigure 6.5b shows the zoomed in version of the same results.



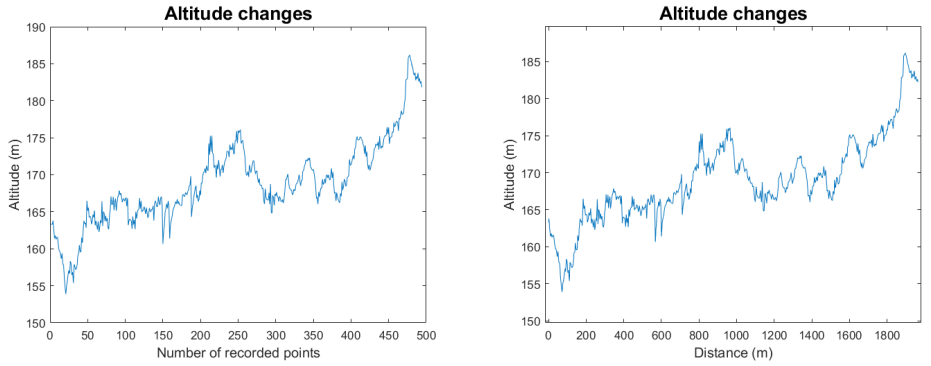
(a) Route map of recorded and cleaned data

(b) Zoomed in route map of recorded and cleaned data

Figure 6.5: Comparing the route map of recorded and cleaned route in metres from the starting position

The overall route is unchanged as shown in subfigure 6.5a. As shown in 6.5b, there are more data points for the same distance travelled on the cleaned route as compared to the recorded route. The last recorded point in the route is ignored if it is not approximately 1 m away from the current intrapolated point. This does not affect the overall route. Importantly, the distance between each waypoint on the route is approximately 1 m. The pace planner will receive reliable and consistent route data regardless of the speed it the recorder was moving at when recording the route.

The elevation of the recorded data and the cleaned data is shown in subfigure 6.6a and subfigure 6.6b respectively.



(a) Recorded data elevation

(b) Cleaned data elevation

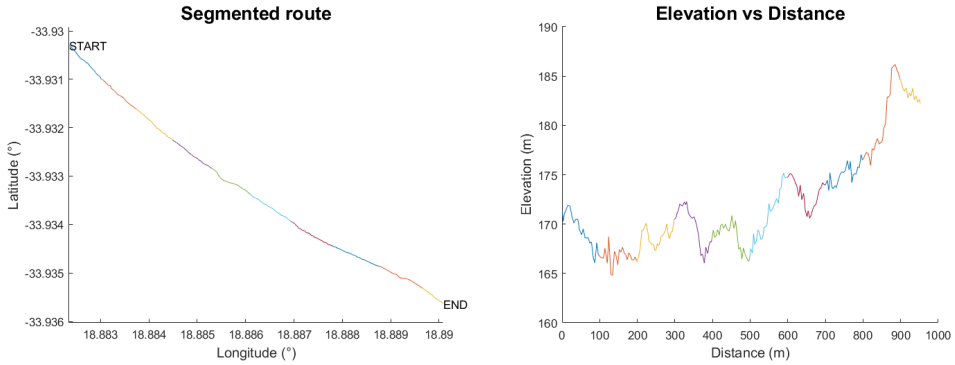
Figure 6.6: Comparing elevation of recorded and cleaned data

The general trend of the altitude is retained. Subfigure 6.6b shows the altitude as a function of distance for the cleaned route. The total distance is 1964 m. Because the distance between each interpolated point is 1 m, the number of altitude points represents the total route distance. The cleaned data has 1964 altitude points compared to 494 altitude points of subfigure 6.6a. The only difference is that the cleaned data elevation has more elevation points to corresponds to the increased coordinate points as expected.

6.3.2. Pace Planner

This section discusses how the pace planner is tested. The pace planner separates the route into different segments and assigns paces on those segments as a function of elevation on the segment. The pace planner will be tested observing how it segments a route, and the paces it assigns to the segments of the route.

To observe how the pace planner segments the route, the pace planner is given a route that is approximately 1 km with elevation changes. The f_{pc} is set to 100. This means that the segments are expected to be 100 m in distance. The segments of the route are shown in Figure 6.7.

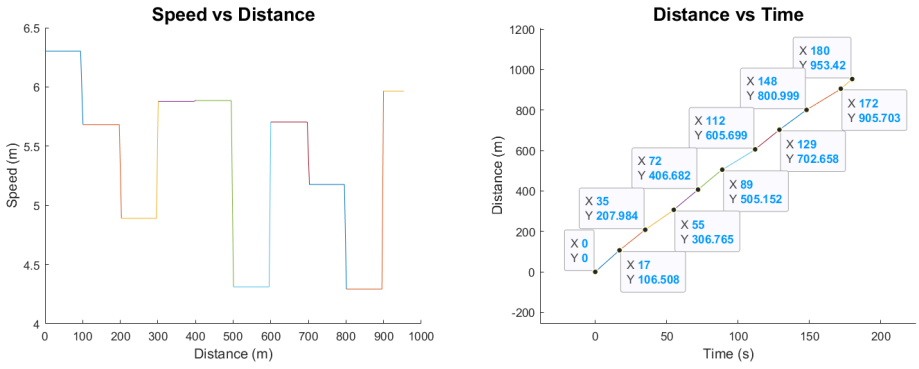


(a) Segmented Route **(b)** Elevation on each segment

Figure 6.7: Route segments

Subfigure 6.7a shows the the latitude and longitude coordinates of the segmented route. The different segments are shown with different colours. Subfigure 6.7b shows the different elevations on each segment. The segments are all approximately 100 m with the exception of the last as expected.

The effect of elevation on the pace is tested. This is done by using the same route as above and setting the desired completion time to 3 minutes (180 seconds).



(a) Speed on each segment **(b)** Distance over time

Figure 6.8: Speed and time after each segment

The colour difference represents the different segments on both subfigures 6.8a and 6.8b. The pace planner's completion time is 180 seconds. This matches the set desired completion time. Subfigures 6.8a shows the adjusted speeds per segment. The segment distances remain 100 m but the speed on each segment is adjusted as a function of the elevation change on the segment. For a positive gradient, the speed on a segment decreases and for a negative gradient, the speed on a segment increases. For segments with very small differences in the final and initial elevation, the speeds on those segments is closer to the new average speed of the whole route than the ones with bigger elevation changes on them.

The pace planner is working as intended. It is capable of segmenting a route as designed and producing paces adjusted to with elevation on each segment. The completion time is also as the user sets.

6.3.3. Virtual Pacer

Different experiments were conducted to test the effectiveness of the virtual pacer. The ability of the virtual pacer to complete the in the desired completion time is tested. Another test is performed to determine whether a human runner can follow the virtual pacer in real time and finish a run in the desired completion time. The speed of the virtual pacer is tested to verify that the paces set by the pace planner are followed. A test to verify that the virtual pacer's distance and location is updated as a function of elapsed time since the start of the race is performed. This is to ensure that the human runner will get the position updates of the pacer and indication of whether they are on pace or not.

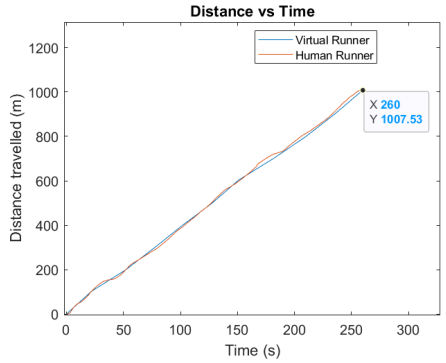
Experiment 1: Completion time

The ability of the virtual pacer to complete the race in the desired time is tested. The virtual pacer was given a route map (latitude, longitude and elevation), the cumulative distance per second, the distance per segment, the speed per segment and a dummy runner's location update every second. A dummy runner is used to simulate what the output paces would be for a completely flat course. The desired completion time was set to 60 seconds with an f_{pc} value of 100.

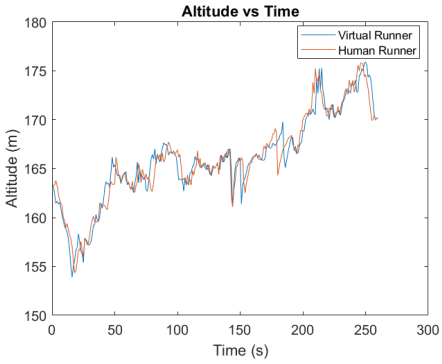
Experiment 2: Completion time for a human runner correctly following the virtual pacer

This test is performed to determine whether a human runner can follow the virtual pacer in real time and finish a run in the desired completion time. For this test, the android application that was written is used to store the virtual pacer's position for every second of the run. For every second, the position of the virtual pacer is displayed on the screen for

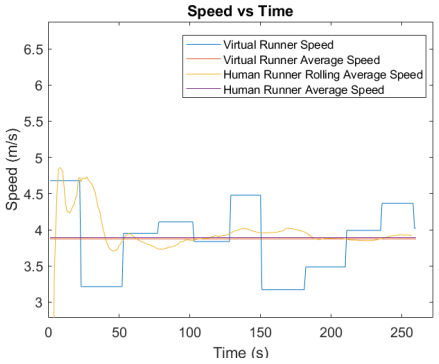
the human runner to follow. The position (latitude and longitude) of the human runner is recorded and stored on the mobile device's memory. The human runner's position data is analysed using Matlab. The desired completion time was set to 260 seconds with an f_{pc} value of 100.



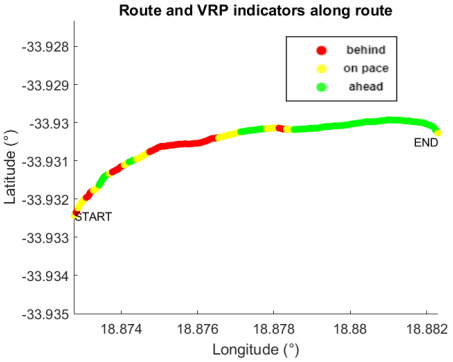
(a) Distance ran by Robot and Runner vs Time



(b) Robot and runner's altitude



(c) Corresponding Robot Speed



(d) Virtual pacer's location profile

Figure 6.9: Runner and pacer comparison

(TALK LOST TIME WHEN DOING COMPARISON AND GOOGLE UPDATE LATENCY. REFERENCE MATLAB. CALIBRATION METHOD)

6.4. Discussion

All the results show that the VRP is capable of enabling a runner to finish at a set completion time. If the runner follows the pacer, they will be able to complete their runs most optimally. Speed is changed per segment depending on the gradient on the segment. Whenever the runner deviates from the pacer, the pacer provides an indication to alert the runner that they are falling behind or running too quickly and pulling away.

Chapter 7

Conclusion

7.1. Overall System Summary and Conclusion

This is what we tried to do, overview of report. list objectives, used matric to get objective. So what?? What do we take from this (chapter 1 background and problem statement) People will be able to pace their runs better for optimal pace and run faster with even effort.

7.2. Shortcomings

1. Algorithm relies on correctness of chosen running technique. When pacing technique changes, algorithm needs to change.
2. Controller limitations (https://www.mathworks.com/help/robotics/ug/pure-pursuit-controller.html).
3. With regards to integrating the algorithm with other fitness applications, the integration process relies heavily on the willingness on the the companies to host the algorithm on their applications regardless of its potential or benefits.

7.3. Recommendation For Future Work

1. Weather affects pacing. Look at weather predictions and adjust pace accordingly.
2. Running surface adjustments.
3. The developed VRP is reliant on the user's ability to follow the given cues and fitness of the runner to follow the recommended pace. There is no extra support for the runner if they deviate too much from the paces. In the future, the model can be designed in a way that allows for the planned paces to change on upcoming segments to allow for lost time to be made up in the most optimal way possible.
4. Because only the distance travelled is compared, if a runner deviates significantly from the route, the virtual pacer will show the deviation but the indicators will still run as if the runner is on the running course.

Bibliography

[1] "WaveLight. feel the pace," https://wavelight-technologies.com/, accessed: 2022-05-02.

[2] E. Advice, "How to pace your run," https://www.rei.com/learn/expert-advice/how-to-pace-your-run.html, accessed: 2022-05-02.

[3] READY.SET.MARATHON, "The marathon pacer – should you run with one?" https://readysetmarathon.com/the-marathon-pacer-do-you-need-one/, accessed: 2022-05-02.

[4] Graymatter, "Cori Cycle," https://www.sciencedirect.com/topics/biochemistry-genetics-and-molecular-biology/cori-cycle, accessed: 2022-05-02.

[5] ——, "Runner's World. how to eat during long runs," https://www.runnersworld.com/nutrition-weight-loss/a20794621/how-to-eat-during-long-runs/, accessed: 2022-05-02.

[6] H. van Dijk. Ron van Megen, "The secret of running," vol. 1, no. 6, p. 68, 2017.

[7] "Stryd. most runners train the wrong intensity," https://buy.stryd.com/gl/en, accessed: 2022-05-02.

[8] Graymatter, "Springer Link. describing and understanding pacing strategies during athletic competition," https://link.springer.com/article/10.2165/00007256-200838030-00004, accessed: 2022-05-02.

[9] ——, "Beatbot. the future faster," https://www.10xbeta.com/puma-beatbot, accessed: 2022-05-02.

[10] "Ghost Pacer. the ghost pacer," https://ghostpacer.com/, accessed: 2022-05-02.

[11] "Zombie,Runs!" https://zombiesrungame.com/, accessed: 2022-05-02.

[12] S. Kettle, "Distance on a sphere: The haversine formula," https://community.esri.com/t5/coordinate-reference-systems-blog/distance-on-a-sphere-the-haversine-formula/ba-p/902128#:~:text=All%20of%20these%20can%20be,longitude%20of%20the%20two%20points., accessed: 2022-05-02.

[13] M. type Scripts, "Calculate distance, bearing and more between latitude/longitude points," https://www.movable-type.co.uk/scripts/latlong.html, accessed: 2022-05-02.

[14] "MathWorks mobiledev," https://www.mathworks.com/help/supportpkg/mobilesensor/ug/mobiledev.html, accessed: 2022-05-02.

[15] GrayMatters, "Android Police. google's latest android version distribution numbers show 11 in dead heat with 10," https://www.androidpolice.com/googles-latest-android-version-distribution-numbers-show-11-in-dead-heat-with-10/, accessed: 2022-05-02.

Appendix A

Project Planning Schedule

Table A.1: Baseline Project Planning Schedule

Period	Planned Work
14/02/22 - 18/02/22	Admin project planning
21/02/22 - 04/03/22	Project objectives and Literature Review
07/03/22 - 08/03/22	Initial report structure
07/03/22 - 06/06/22	Report Write-up
09/03/22 - 11/03/33	Collect and test route data samples
14/03/22 - 08/04/22	Write the recorder, pace planner and virtual pacer software
11/04/22 - 22/04/22	Test the hardware and software
25/04/22 - 25/05/22	Reiterate if needed
25/04/22 - 25/05/22	Write android application and reiterate if needed
30/05/22 - 06/06/22	Report feedback and editing
06/06/22	Report submission
07/06/22 - 13/06/22	Video and poster and presentation

Table A.2: Adjusted Project Planning Schedule

Period	Planned Work
14/02/22 - 18/02/22	Admin and project planning
21/02/22 - 04/03/22	Project objectives and Literature Review
07/03/22 - 08/03/22	Initial report structure
07/03/22 - 13/06/22	Report Write-up
09/03/22 - 11/03/33	Collect and test route data samples
14/03/22 - 08/04/22	Write the recorder, pace planner and virtual pacer software
11/04/22 - 22/04/22	Test the hardware and software
25/04/22 - 03/06/22	Reiterate if needed
25/04/22 - 03/06/22	Write android application and reiterate if needed
07/05/22 - 13/06/22	Report feedback and editing
13/06/22	Report submission
13/06/22 - 18/06/22	Video and poster
20/06/22	Oral presentation

Appendix B

ECSA Outcomes Compliance

B.1. Problem solving

ELO 1. Problem solving: Identify, formulate, analyse and solve complex engineering problems creatively and innovatively.

A complex and ill-defined engineering problem of developing a virtual running pacesetter is identified and a problem statement and project objectives are formulated in chapter 1. The problem is analysed, understood and solved in Chapter 2 through to 6. In Chapter 2, a variety of sources were use to understand the problem and develop a solution. Different pacing strategies employed by human runners and human pacesetters are considered, a running model is sourced from literature, and an optimal running technique that adjusts for elevation changes is selected. In Chapters 3 to 5 the system overview, detailed design and implementation are described. In Chapter 5 experiments are performed to verify the solution.

The project represents a complex engineering problem since a virtual running pacer for road and trail running is a complex and ill-defined problem. The problem includes several sub-problems, including recording and cleaning the running route, pace planning taking into account elevation changes, and executing the virtual pacer during a run. The solution is not obvious and requires originality.

B.2. Application of scientific and engineering knowledge

ELO 2. Application of scientific and engineering knowledge: Apply knowledge of mathematics, natural sciences, engineering fundamentals and an engineering speciality to solve complex engineering problems.

Scientific and engineering knowledge were applied to model and solve the complex engineering problem of developing a virtual running pacer. In Chapter 2 a biological concept in form of the cori cycle is presented and the solution is based on what is understood

to happen when human runners run with intensity. A mathematical model for a runner's output power that takes into account elevation changes was sourced from literature. In Chapter 5 software algorithms were developed and implemented to pre-process the raw GPS positions measurements, to segment the route, and to plan the pace for each segment taking into account elevation changes. Mathematical transformations were used to transform GPS measurements from latitude and longitude coordinates to positions and distance in meters in Chapter 4 to 6. Mathematical analyses were used to analyze the test results.

B.3. Engineering design

ELO 3. Engineering Design: Perform creative, procedural and non-procedural design and synthesis of components, systems, engineering works, products or processes.

The problem is not frequently encountered hence the solution is not a build-up from related work but requires an original design. A virtual running pacesetter was designed in Chapters 3 to 4. In Chapter 3, the system was conceptually designed to comprise of a route recorder, a pace planner, and a virtual pacer. In Chapter 4, the detailed design of each component was performed. The detailed design consisted of algorithm design and software design. In Chapter 5, the components were integrated and implemented on a mobile device. An iterative approach is taken in solving encountered subproblems and in eliminating various alternatives which were not the best solutions. Experiment design procedures were undertaken in Chapter 6 to test components of the virtual running pacesetter.

B.4. Investigation, experiments and data analysis

ELO 4. Investigations, experiments and data analysis: Demonstrate competence to design and conduct investigations and experiments.

In Chapter 2, different pacing strategies employed by human runners and human pacesetters are investigated. Research is performed to source a running model from literature, and to find an optimal running technique that takes elevation changes into account. Data analysis is done to establish the data flow of the virtual running pacesetter system. From Chapter 3 to 6, decisions are made on how data is received, unwrapped, and packaged for subsequent steps. In Chapter 6, experiments are designed and conducted to test the GPS sensor, recorder, pace planner, virtual pacer and to test the system as a whole.

B.5. Engineering methods, skills and tools

ELO 5. Engineering methods, skills and tools, including Information Technology: Demonstrate competence to use appropriate engineering methods, skills and tools, including those based on information technology.

Competence to use appropriate engineering methods, skills, and tools was demonstrated. The virtual running pacesetter was developed using the Matlab programming and numeric computing environment. The Matlab Mobile application was used to record GPS position measurements using an android mobile device. The route recorder and pace planner were implemented using Matlab scripts. The virtual pacer was implemented using Matlab Geoplayer and a new written android application using google map services. Matlab was also used for simulation, testing and reporting of the experimental results. The Matlab online cloud, Git and GitHub were used to store project information.

B.6. Professional and technical communication

ELO 6. Professional and technical communication: Demonstrate competence to communicate effectively, both orally and in writing, with engineering audiences and the community at large.

Competence to communicate effectively in writing is demonstrated through this written project report and its successful completion. The oral and poster presentation demonstrate competence to orally communicate effectively.

B.7. Individual work

ELO 8. Individual work: Demonstrate competence to work effectively as an individual.

Competence to work effectively as an individual was demonstrated. The conceptualisation of the virtual running pacesetter was done individually by myself to improve the sport that I love. The development, design, implementation and reporting was performed by myself with the guidance of my academic supervisor and it was a complete successful.

B.8. Independent Learning Ability

ELO 9. Independent Learning Ability: Demonstrate competence to engage in independent learning through well-developed learning skills.

Competence to engage in independent learning was demonstrated throughout the project. The project required new knowledge of pacing strategies for track, road, and trail running, a running model, and an optimal pacing strategy that takes into account elevation adjustments. The project also required new knowledge of mathematical coordinate transformations and algorithms to perform route cleaning and pace planning. In Chapter 5 learning skills are showcases as android application development is not part of the standard Electrical and Electronics modules. Unfamiliar Matlab software packages were learnt and used in Chapters 5 and 6. The new knowledge was acquired through literature reviews and online tutorials.

Appendix C

Problems and solutions

1. Finding a reliable source of running data. First option was to use a running website to collect running data. Strava was considered but it turns out that to get any data on strava, a query is sent and a token is given. Then the token is used to get the data but the token expires. Google maps also has a similar system for elevations data but uses a code instead. Using a compact external GPS device was considered. Matlab mobile has a sensor option which allows users to collect data using sensors of a phone. This was the chosen option because of its simplicity and compatibility with matlab.
2. Matlab uses API 30. 3. Talk about extrapolation+its+test

Notes

1. Effort should be the same, not pace. Runners try to keep the same pace throughout runs and neglect elevation changes. On higher elevation gain sections of a run, they exert more effort and burn more lactic acid than then should to keep the same pace.
2. "You are behind when you are more than a running speed distance behind". If the running speed is 3m/s then you are only behind if you are more than 3m behind the robot.

Appendix D

Lessons Learnt

1. Biology, the cori cycle. 2. APIs, android app dev. 3. Research, as per chap2. 4. Map data manipulation.

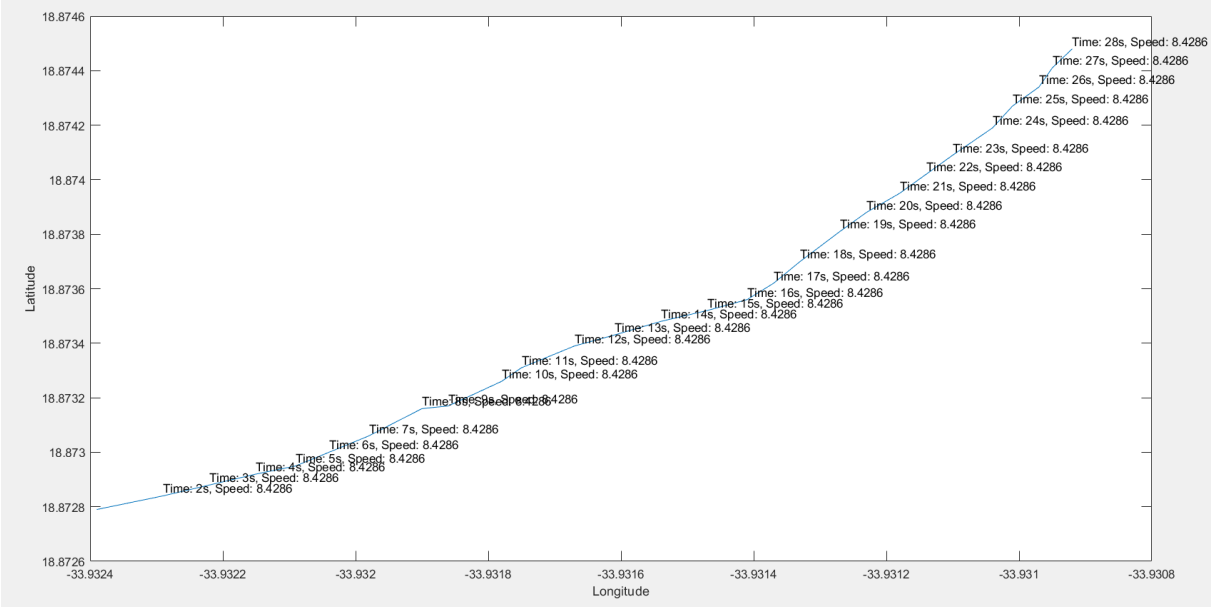
Appendix E

Code Used

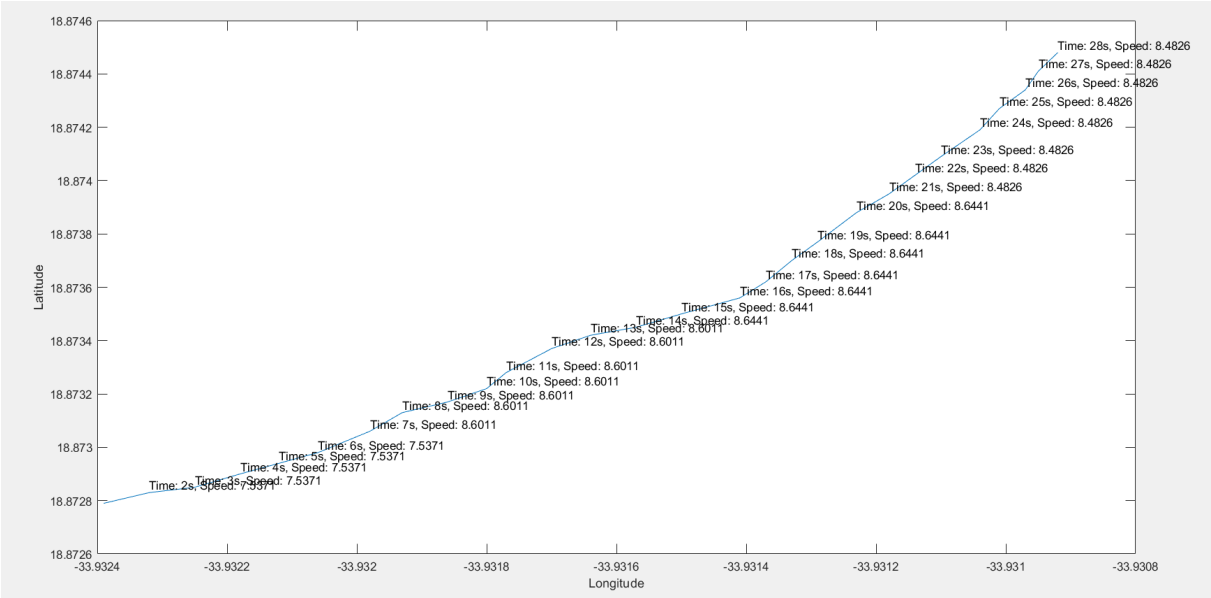
Ghost Pacer YouTube: https://www.youtube.com/watch?v=wKEW$_c$6$NEIU$

See below. For perfect input data; data that allows for segments to be separated by time and have equal distance, the algorithm to calculate the average distance for each segment behaves as intended as illustrated below.

```
avgspeed = cat(1,avgspeed,(59)/nww(counter)); % assuming equal segment distances
                                              % shows algorithm works (perfect data input)
```



More significantly, the algorithm automatically accounts for the inevitability of having imperfect data. When the distance is not the same for all segments, the algorithm sets different average running speeds for each segment depending on the distance to be ran on that segment, regardless of the existence of change in elevation. This is illustrated below.

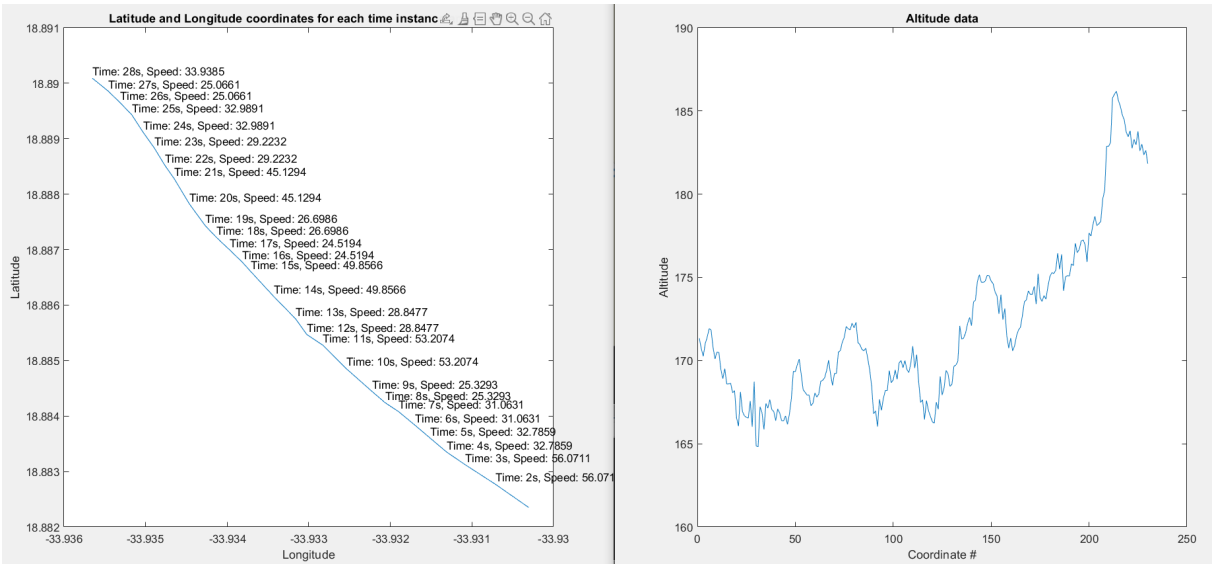


There is a time error introduced by the following line of code,

```
k = k + 1; % final time is almost x; this line takes 0.3 ms to execute
```

It is only executed when the robot model is not supposed to move. With the frequency of pace changes fixed at 15 input points, the time error is observed to be 0.3ms/second as shown below. The effect of this is equivalent to only 4.32s in a 4 hour marathon and a mere 0.36s for a 20 minute 5 km. Thus the introduced time error was considered to be negligible and tolerated. The total time error is shown below to be 30ms/second. This is from the runner's current position being fetched and plotted and other checks. A 43.2s error for a 4 hour marathon is significant. This error can be corrected by subtracting it from the sampling time that is fed to rate control inside the loop.

Below is an illustration of the speed changes on the left with regard to the elevation changes on the right. The results show that the speed is increased as the altitude drops and increased as the altitude rises as expected.



```
function [df, dInc] = checknll(lat,long,size)
    R = 6371000;
    df = zeros(1);
    dInc = [];

    for i=2:1:size
        a11 = (lat(i)-lat(i-1))*pi/180;
        a41 = (long(i)-long(i-1))*pi/180;

        a1 = sin(a11/2)^2;
        a2 = cos(lat(i-1)*pi/180);
        a3 = cos(lat(i)*pi/180);
        a4 = sin(a41/2)^2;

        a = a1 + a2 * a3 * a4;
        c = 2 * atan2(sqrt(a),sqrt(1-a));
        d = R * c ; % in meters

        df = cat(1, df, df(i-1) + d); % in meters
        dInc = cat(1, dInc, d);
     end
end
```

Figure E.1: Distance calculation

```
function [df, dInc, NewLatLong, NewAlt] = distanceIncr(lat,long,alt,course,size)
   R = 6371000;
    df = zeros(1);
    dInc = [];
    NewLatLong = [lat(1) long(1)];
    NewAlt = [alt(1)];
    d_move = 1;

    for i=2:1:size
        latim1temp = lat(i-1);
        longim1temp = long(i-1);

        a11 = (lat(i)-latim1temp)*pi/180;
        a41 = (long(i)-longim1temp)*pi/180;

        a1 = sin(a11/2)^2;
        a2 = cos(latim1temp*pi/180);
        a3 = cos(lat(i)*pi/180);
        a4 = sin(a41/2)^2;

        a = a1 + a2 * a3 * a4;
        c = 2 * atan2(sqrt(a),sqrt(1-a));
        d = R * c ; % in meters

        df = cat(1, df, df(i-1) + d); % in meters
        dInc = cat(1, dInc, d);

        br = (course(i))*pi/180;
        lat1 = latim1temp*pi/180;
        long1 = longim1temp*pi/180;
        dcheck = 0;
        while(dInc(i-1)-dcheck>d_move)
            lat2 = asin(sin(lat1)*cos(d_move/R) + cos(lat1)*sin(d_move/R)*cos(br));
            long2 = long1 + atan2(sin(br)*sin(d_move/R)*cos(lat1),cos(d_move/R)-sin(lat1)*sin(lat2));
            NewLatLong = cat(1, NewLatLong, [lat2*180/pi long2*180/pi]);
            NewAlt = cat(1, NewAlt, (alt(i)-alt(i-1))/dInc(i-1)*dcheck+alt(i-1));
            y = sin(long2-long1)*cos(lat2);
            x = cos(lat1)*sin(lat2) - sin(lat1)*cos(lat2)*cos(long2-long1);
            br = atan2(y, x);
            long1 = long2;
            lat1 = lat2;
            dcheck = dcheck + d_move;
        end
    end
end
```

Figure E.2: Increasing the course resolution

```
strava = load('set16el.mat'); % loads matlab sensor data
strava2 = load('set12el.mat');
strava3 = cat(2,strava,strava2);

p1 = strava.Position;
p2 = strava2.Position;
Position = cat(1,p1,p2);

lat = Position.latitude;
long = Position.longitude;
alt = Position.altitude;
course = Position.course;
size = length(lat);

[df, dInc, nll, new_alt] = distanceIncr(lat,long,alt,course,size);
[newdf, newdInc] = checknll(nll(:,1),nll(:,2),length(nll(:,1)));

plot(1:length(alt),alt);
xlabel('Number of recorded points');
ylabel('Altitude');
title('Altitude changes');
figure
plot(1:length(new_alt),new_alt);
xlabel('Number of exprapolated points');
ylabel('Altitude');
title('Altitude changes');

player = geoplayer(lat(1),long(1),21);
player.Parent.Name = 'RecorderHD';
player.Basemap = 'streets';
plotRoute(player,nll(:,1),nll(:,2),"Color","red");
plotRoute(player,lat,long,"Color","blue","LineWidth",1);
for i = 1:length(nll(:,1))
    plotPosition(player,nll(i,1),nll(i,2),"TrackID",1,"Marker","*","Label","HD");
end
```

Figure E.3: Clean data testing

newdInc

1097x1 double

	1
1	1.0000
2	1.0000
3	1.0000
4	1.0000
5	1.0000
6	0.9687
7	1.0000
8	1.0000
9	1.0000
10	0.9654
11	1.0000
12	1.0000
13	1.0000
14	1.0000
15	0.9746
16	1.0000
17	1.0000
18	1.0000
19	0.9198
20	1.0000
21	1.0000
22	1.0000
23	1.0000
24	1.0484
25	1.0000

Figure E.4: Resulting distance between subsequent coordinates

```
function [df, distance_per_segment] = distanceIncr3(lat,long,size,fpc)
    R = 6371000;
    step = 1;
    distance_per_segment = [];
    df = zeros(1);
    for i=2:1:size
        a11 = (lat(i)-lat(i-1))*pi/180;
        a41 = (long(i)-long(i-1))*pi/180;

        a1 = sin(a11/2)^2;
        a2 = cos(lat(1)*pi/180);
        a3 = cos(lat(size)*pi/180);
        a4 = sin(a41/2)^2;

        a = a1 + a2 * a3 * a4;
        c = 2 * atan2(sqrt(a),sqrt(1-a));
        d = R * c ; % in meters

        df = cat(1, df, df(i-1) + d) ; % in meters

        if mod(i,fpc)==0
           distance_per_segment = cat(1,distance_per_segment,df(i)-df(step));
           step = i; % the end point (index) of recorded elevation
        end
    end
    if mod(i,fpc)~=0
       distance_per_segment = cat(1,distance_per_segment,df(size)-df(step));
    end
end
```

Figure E.5: Calculating course distance and distance per segment

```
function newspeed = segmentTimes(avgspeed, alt, fpc, dps, tdt)
    size = length(alt);
    elevation = [];
    step = 1;
    for i = fpc:1:size
      if mod(i,fpc)==0
        elevation = cat(1,elevation,alt(i,1)-alt(step,1));
        step = i; % the end point (index) of recorded elevation
      end
    end
    if mod(i,fpc)~=0
       elevation = cat(1,elevation,alt(size,1)-alt(step,1));
    end

    newspeed = [];
    televation = sum(abs(elevation),'all'); % total elevation
    if televation==0
        for i=1:length(elevation)
            newspeed(i) = avgspeed;
        end
    else
        elevation = elevation/televation; % gets the % elevation contribution of each segment
        elevation = avgspeed/(1+elevation); % assigns speed each segment
        newspeed = tdt/new_completion_time(dps, elevation); % new average speed
        ratio = avgspeed/newspeed; % ratio (expansion or compression of average speed)
        newspeed = ratio*elevation; % adjusted average speed for each segment
    end
end
```

Figure E.6: Calculating new speeds for each segment

```
function new_completion_time = new_completion_time(dps, new_avgspeed_on_segment)
    new_completion_time = [];
    naos = new_avgspeed_on_segment;
    for i = 1:length(dps)
        new_completion_time = cat(1,new_completion_time,dps(i)/naos(i));
    end
    new_completion_time = sum(new_completion_time,'all');
end
```

Figure E.7: Consequential new completion time

```
function dist = diste2(x,speed_per_segment,distance_per_segment,df, size)
    dist = zeros(1);
    step = 1;
    distance_to_travel = distance_per_segment(step);
    for i=2:1:x+1
        if dist(i-1) >= distance_to_travel
            step = step + 1;
            distance_per_segment(step);
            distance_to_travel = distance_to_travel + distance_per_segment(step);
        end
        dist = cat(1,dist,dist(i-1)+speed_per_segment(step));
    end
end
```

Figure E.8: Translating speed per segment to distance per second

```
st = 1;
player = geoplayer(oldlat(1),oldlong(1),18);
player.Parent.Name = 'Runner vs robot';
player.Basemap = 'streets';
plotRoute(player,oldlat,oldlong,"Color","cyan");
sampleTime = rateControl(st); % 1 second sampling rate to represent real world
```

Figure E.9: Visualiser setup

The dummy data in figure E.10 is the runner at the speed while the data was recorded and the robot data is the output of the controlled speed.

```
while(i<=sizemax && play==true)
    if i<size
        plotPosition(player,oldlat(i),oldlong(i),"TrackID",1,"Marker","+","Label","Dummy");
    end

    if i<=x-1
        if dist(i) <= df(k)
            if i<=size
                new_alt = cat(1, new_alt, alt(k));
            end
            i = i + 1;
            results = cat(1,results, [oldlat(k) oldlong(k)]); % coordinates at this time instance
            plotPosition(player,oldlat(k),oldlong(k),"TrackID",i,"Marker","*");

            if dist(i) >= distance_to_travel
                distance_to_travel = distance_to_travel + distance_per_segment(step);
                step = step + 1;
            end

            speed{i} = speed_per_segment(step);

            current_speed = speed_per_segment(step)
            distanceRan = dist(i)
            TimeElasped = sampleTime.TotalElapsedTime
            waitfor(sampleTime); % wait for 1 second
        else
            k = k + 1;
        end
    else
        i = i + 1;
        plotPosition(player,oldlat(size),oldlong(size),"TrackID",i,"Marker","*","Label","RobotDone");
        % waitfor(sampleTime); % turn this on
    end

    if i==x
        new_alt = cat(1, new_alt, alt(size));
        results = cat(1,results, [oldlat(size) oldlong(size)]);
        plotPosition(player,oldlat(size),oldlong(size),"TrackID",i,"Marker","*","Label","RobotDone");
        speed{i} = speed_per_segment(step);
        current_speed = speed_per_segment(step)
        distanceRan = df(size)
        TimeElasped = sampleTime.TotalElapsedTime
        completionTime = round(TimeElasped)
        status = 'Run must have been completed already'
    end
end
```

Figure E.10: Visualiser with dummy data

```
load('testset.mat');
lat = Position.latitude;
long = Position.longitude;
alt = Position.altitude;
course = Position.course;
size = length(lat);
scatter(lat ,long , [] , linspace(lat(1),lat(size),size) ,"filled");
title('Single point test');
xlabel('Longitude');
ylabel('Latitude');

u = sum(alt,"all")/size;
ud = 0;
sd = ones(size,1);
for i = 1:size
    ud = ud + (alt(i)-u)^(2);
end
sd = sd*(sqrt(ud)/sqrt(size));


figure
errorbar(0:size-1,alt,sd)
title('Altitude test');
xlabel('Number of collected points');
ylabel('Altitude');
```

Figure E.11: Hardware testing

(a) Staying on a marked spot for the static tests

Position

25x6 timetable

Timestamp	1 latitude	2 longitude	3 altitude
15-May-2022 08:4...	-33.9323	18.8724	159.5970
15-May-2022 08:4...	-33.9323	18.8724	159.5970
15-May-2022 08:4...	-33.9323	18.8724	159.5970
15-May-2022 08:4...	-33.9323	18.8724	159.5980
15-May-2022 08:4...	-33.9323	18.8724	159.5970
15-May-2022 08:4...	-33.9323	18.8724	159.5970
15-May-2022 08:4...	-33.9323	18.8724	159.5960
15-May-2022 08:4...	-33.9323	18.8724	159.5940
15-May-2022 08:4...	-33.9323	18.8724	159.5930
15-May-2022 08:4...	-33.9323	18.8724	159.5920
15-May-2022 08:4...	-33.9323	18.8724	159.5890
15-May-2022 08:4...	-33.9323	18.8724	159.5880
15-May-2022 08:4...	-33.9323	18.8724	159.5850
15-May-2022 08:4...	-33.9323	18.8724	159.5850
15-May-2022 08:4...	-33.9323	18.8724	159.5850
15-May-2022 08:4...	-33.9323	18.8724	159.5830
15-May-2022 08:4...	-33.9323	18.8724	159.5740
15-May-2022 08:4...	-33.9323	18.8724	159.5770
15-May-2022 08:4	-33.9323	18.8724	159.5770

(b) Hardware test set

```
load('dt3.mat');
lat = Position.latitude;
long = Position.longitude;

player = geoplayer(lat(1),long(1),21);
player.Parent.Name = 'Runner vs robot';
player.Basemap = 'streets';
plotRoute(player,lat,long,"Color","cyan");

figure
scatter(lat(5),long(5));
hold on;
scatter(lat(50),long(50));

load('dt5.mat');
lat = Position.latitude;
long = Position.longitude;
plotRoute(player,lat,long,"Color","black");

scatter(lat(5),long(5));
scatter(lat(50),long(50));

load('dt1.mat');
lat = Position.latitude;
long = Position.longitude;

scatter(lat(5),long(5));
scatter(lat(51),long(51));

load('dt2.mat');
lat = Position.latitude;
long = Position.longitude;

scatter(lat(5),long(5));
scatter(lat(50),long(5));

load('dt4.mat');
lat = Position.latitude;
long = Position.longitude;

scatter(lat(5),long(5));
scatter(lat(45),long(45));

title('Latitude and Longitude at start and end');
xlabel('Longitude');
ylabel('Latitude');
legend('start1', 'end1', 'start2', 'end2');
```

Figure E.13: Repeatability test