Development and design of an autonomous/non-human running pacer

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

May, 2022

Acknowledgements

I would like to thank my dog, Muffin. I also would like to thank the inventor of the incubator; without him/her, I would not be here. Finally, I would like to thank Dr Herman Kamper for this amazing report template.
I will forever appreciate Dr Engelbrecht for agreeing to walk down this path with me. I cannot begin to express my thanks for his guidance and being my source of calm when I got confused.

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*	May, 2022 Datum / *Date*

Abstract

English

The English abstract.

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.

Afrikaans

Die Afrikaanse uittreksel.

Contents

List of Figures

List of Tables

Nomenclature

Variables and functions

Host app	Strava, Zepp Life or standalone pacer app ect
P	Runner's Power (W)
P_r	Running resistance
P_a	Air resistance
P_c	Climbing resistance
m	Runner's mass (kg)
v	Runner's speed ($\mathrm{m\,s^{-1}}$)
c	ECOR ($\mathrm{kJ.kg^{-1}}.km^{-1}$)
p_{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
δ	Angular distance
ϕ	Latitude
ψ	Longitude
P_s	Specific Power ($\mathrm{W\,kg^{-1}}$)

Acronyms and abbreviations

APP	Application
ECOR	Energy Cost Of Running
GPS	Global Positioning System
4IR	Fourth Industrial Revolution
AR	Augmented Reality
ARP	Autonomous Running Pacer

Chapter 1

Introduction

1.1. Background

Why does an autonomous running pacer matter? Broad statement.
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 similar solution is used in road or cross country running. The 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" and "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. This often happens when a runner is prepared to run at a constant steady pace throughout their run and encounters a 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. Every runner knows this but still, most runner 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 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 creating an autonomous running pacer to enable runner of all levels to run optimally paced runs specifically during road and cross country training and racing.

1.2. Socio-economic Implications

With the adoption of 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 autonomous pacer which this project will aim to develop. The autonomous pacer of this project can be classified as automation. It aims at automating and significantly improving a task which has been predominately performed by other human beings. Most major running events have human pacers doing

what the autonomous pacer will partly be doing. In elite races, these human pacers 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 pacers even during amateur events. It shift in mindset and breaking of the status quo is necessary for the success of technologies such as the autonomous pacer. Nothing but the benefits of the autonomous pacer can be 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 improving the sport's reach.
- Enables new markets and research opportunities around the sports of running, in turn enabling runners to make more money from running. This includes the human pacers, new and older runners.
- Finally, SOMETHING IMPORTANT WAS ON MY MIND, NOW ITS GONE LOL.

1.3. Problem statement

Having considered the socio-economic implications of the adoption of 4IR technologies, particularly of the autonomous pacer, the project aims to develop and design an autonomous running pacing system for optimal running. The goal is to produce a running pacer capable of providing the user with paces throughout a run which enable them to run with the least amount of effort deviation throughout the run, allowing for maximised performance. The autonomous pacer will allow many runners to train and run races outside of the track more optimally than ever before. It will also be a ready to use tool because of it's independence on runner body specific data, unlike nothing on the market yet. Most training runs are not logged on the track, so the autonomous pacer will secondarily serve as a training optimization tool.

1.4. Objectives

Overall, the developed pacer must return the correct location profile enabling a runner to know where they must be located for a given time instance of a run. This is how the above will be obtained:

- A way to source running route data must be found.
- An autonomous running pacer model must be developed.

- The best pacing technique must be found and deployed to the developed pacer as an algorithm.
- The algorithm must be able to adjust the paces accordingly throughout the run given the changes in elevation.
- The pacer must provide a location profile to enable a runner to complete a given route at a given time (assuming it is a realistic time).

1.5. Summary of work

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 altitude. 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 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 5: A discussion and illustration of how the algorithm can be used practically.
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

2.1. Development

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

2.1.1. Optimal Running Techniques

In the past, an emphasis was placed on even splits - running at the same pace - for road and cross country running. This is shown in the conclusion of the article, "Describing and Understanding Pacing Strategies during Athletic Competition." [2]. An interesting admission in that article reads as follows, "Whether these descriptive findings represent optimal scenarios requires further research." As stated in section 1.1, the running effort required on an even or downhill surface is not the same as the effort required on an uphill for the same pace. Runners tend to try to hit the same pace targets regardless of changes in their immediate surroundings and ultimately build up lactic acid faster than their bodies can get rid of. The required running effort can be quantified as the energy lost for a distance travelled. This is known as the Energy Cost Of Running, $ECOR$. A focus on the $ECOR$ is exceedingly better than a focus on pace for optimal running. As the $ECOR$ increases, the runner must slow down and as the $ECOR$ decreases, the runner must speed up. Maintaining a constant $ECOR$ is the most optimal running technique to ensure runners run perfectly paced runs (here: https://buy.stryd.com/gl/en).

2.1.2. The cori cycle

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 [3], lactate is converted back to glucose - the source of energy production - removing the hydrogen ions. The cori cycle is illustrated on Figure 2.1. "Bonking" occurs when the build up of hydrogen ions or 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

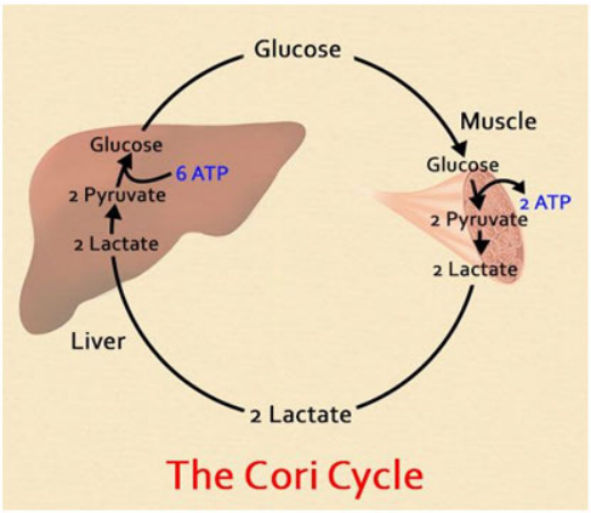


Figure 2.1: The Cori Cycle

will eventually lead to an excessive lactic acid build up as no runner possesses infinite amounts of glucose reserves. For this reason, it is wildly suggested [4] 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.

2.1.3. The Running Model

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

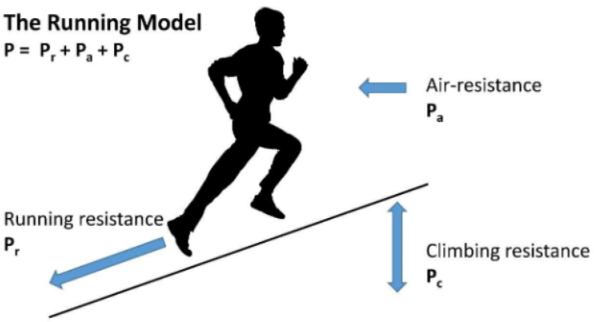


Figure 2.2: The running model

The running model is described in the book "The secret to running" [5]. The runner's power, P, 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 book describes the different powers to be as follows:

- The **running resistance** is a product of a runner's mass, their speed and their ECOR, as shown in equation 2.5 below.

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

  If we let ECOR in 2.5 be c, then,

$$P_r = cmv \tag{2.2}$$

- Equation 2.3 shows the **air resistance** to be a product of the air density, p_{air}, air resistance factor, $c_d A_r$, the square of the the sum of the runner's and wind's speed, and the runner's speed.

$$P_a = \frac{1}{2} p_{air} c_d A_r (v + v_w)^2 v \tag{2.3}$$

- The **climbing resistance** is a product of the gradient, the runner's mass, gravitational acceleration, and the runner's speed as shown in equation 2.4.

$$P_c = imgv \tag{2.4}$$

  The runner's power is,

$$P = cmv + \frac{1}{2} p_{air} c_d A_r (v + v_w)^2 v + imgv \tag{2.5}$$

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; speed not zero when running.

2.2. Related Work

2.2.1. PUMA beatbot and other line following pacerbots

In 2016, Puma launched the Puma beatbot [6]. It is a line following pacing robot capable of guiding a runner along a track at constant pace depending on the time they ant to beat. This type of technology pioneered automating running pacing and since then several replicas have been made. Line following robots are usually connected to a smart phone which is used to input the desired completion time and distance to be travelled. The robot

calculates the constant average running pace. The robot requires to be placed on a line, usually along a running track. It uses cameras, ultrasonic sensors, actuators and motors to follow the line it is placed on while maintaining the calculated pace for the specified distance. The Puma beatbot did this impeccably, 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 robot also does 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 [7] is a pacing tool aimed at allowing runners to have a target to following when aiming for a certain running time. It uses AR headsets to place a augmented robot running at the pace the runner desires besides the runner. Like the technologies discussed in subsection 2.2.1, the Ghost pacer is not robust and is also made from sophisticated and costly hardware. Greatly reducing its accessibility. (INSPIRES NEED FOR FURTHER IMPROVEMENT)

2.2.3. Zombies, Run

Zombies, Run [8] is an interactive fitness app that engages runners into a post apocalyptic story and prompts them to walk jog or run. The app prompts runners to change their speed depending on whether or not they are chased by zombies. The app proves the possibility of pace control for runners and illustrates the extent - how fun and interactive - pace controlling can be made. The app 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 the autonomous pacer. Zombies, Run aimed at making running fun and the autonomous pacers aims at optimising running input vs output. (INSPIRES LIGHT)

2.2.4. Stryd

Stryd [9] is a Footpod that analyzes the movement of a runner's foot and estimate 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. Stryd's usage of a power based matrix 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 thus is runner specific. The autonomous pacer 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)

Chapter 3

System Overview

This chapter introduces and explains the functions of the system's components and how each component interacts with other components. It also briefly discusses alternative ideas which were considered. Figure 3.1 shows the system and its components.

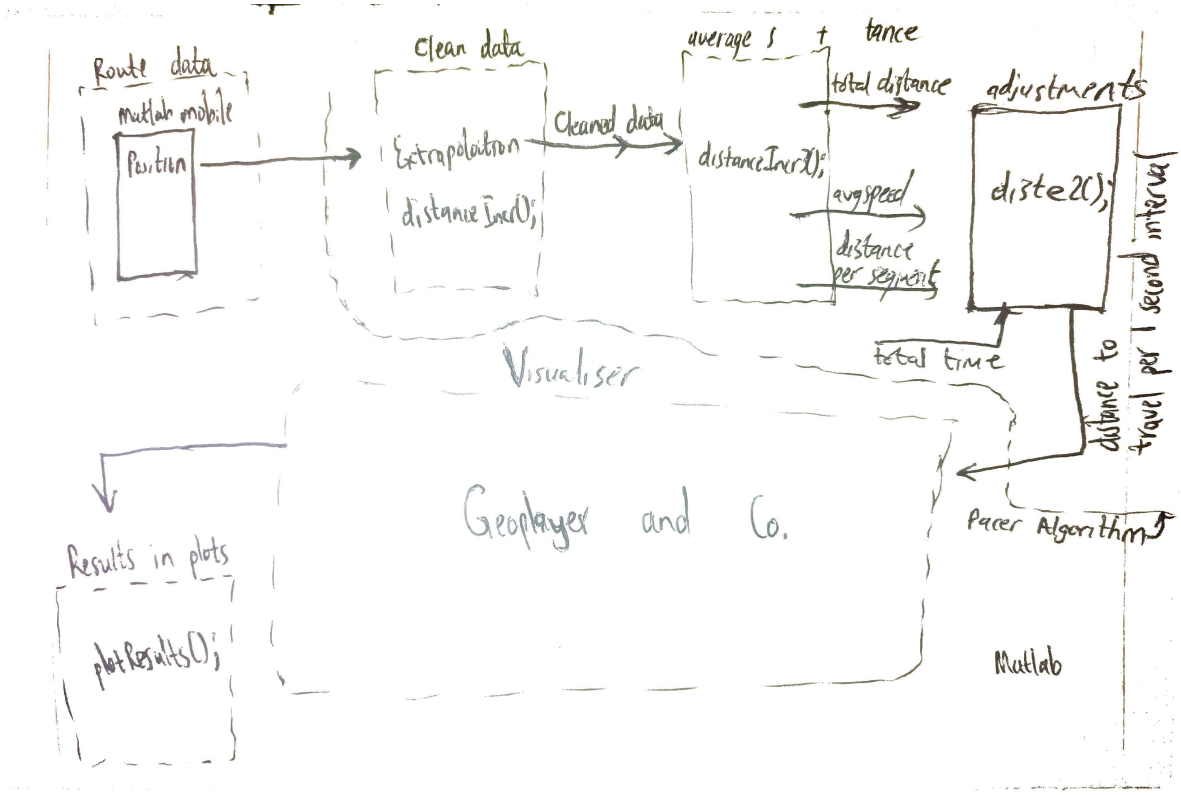


Figure 3.1: System diagram

3.1. Hardware

3.1.1. Android device

Could have used an external device (beatbot/pacebot) but requires more components (compass, gps module), makes the design unnecessarily expensive and complex (do not redesign the wheel), could use have used watch. Why used android device.

The first project objective is to find a way to source running route data. The choice of hardware was made particularly to serve this purpose. Having recognised that route data must be collected and stored for processing, 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. With the realisation that a smartwatch is more naturally a fit for the purpose of the project and its built-in capabilities to collect and store running data, a shift towards using a smartwatch to host the autonomous pacer occurred. Because of the wide usage of phones for fitness activities, a decision to use android devices for the prototype of the autonomous pacer was made. The android device is used in the project to collect and store running route data and send to the algorithm blocks of the pacer.

3.2. Software

3.2.1. Matlab and Matlab Mobile

Matlab was chosen as the platform in which the autonomous pacer will be developed on. The design choice was made because Matlab is geared more towards simulation than other programming platforms. That is what is required for the project as testing will be required in phases multiple of the project. Instead of collecting route data using existing fitness applications or directly google maps, Matlab mobile was the chosen option. This is because it has a sensor option which allows users to collect data using sensors of a phone. This includes position data which the pacer algorithm requires. Although the existing fitness applications use the onboard GPS module as the sensor function on Matlab mobile, Matlab mobile was the chosen option also because of its simplicity and compatibility with Matlab, the chosen development and simulation platform.

3.2.2. Clean data

As the old adage would say, garbage in, garbage out. The recorded route data is susceptible to GPS accuracy. Instead of depending on the data to be accurately recorded, this data cleaning step is performed to ensure that the pacer algorithm is fed reliable and the expected running route data. The resolution of the route can also be controlled allowing for the algorithm to produce movement as accurate as a runner can run. This preprocessing step ensures that movement from one GPS coordinate to the next is unrealistic. Allows for a realistic path to be followed without the need of collecting too many way-points along the route using Matlab mobile.

3.2.3. Distance and average speed calculator

This function block receives the preprocessed route data from the clean data function block and uses it to calculate the distance of the route and average speed required to finish the route given the proposed completion time. It also separates the route into segments of equal distance.

3.2.4. Controller

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 do this is by controlling the energy lost during a run. Using the data from the distance and average speed calculator block, the altitude changes are considered and the speed on each segment is adjusted with respect to the elevation gain or loss on that segment. Then the distances and to be travelled by the runner at every time interval is recorded and passed to the visualiser function blocks along with the pacer's proposed location profile for the run.

3.2.5. Visualiser

A way to see the results of the pacer and a runner's progress is required. The algorithm is designed to help runners visualise how far off they are from the most optimal pace at points in their run. The visualiser acts as a visualisation tool for the results of the algorithm, as a runner would similarly see. Matlab's geoplayer simulator was chosen for the task. 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.

Chapter 4

Detailed Design

This chapter explains how the varies system functional blocks individually work. It investigates the influence of the hardware and software choices on the project final product. It also justifies the choices and mentions mitigation measures taken to reduce potential pacer failure risks.

4.1. Hardware

4.1.1. Built-in GPS

According to the Stryd website, Stryd's motion capture technology measures paces with more precision, consistency and responsiveness than GPS [9]. This further raises question about the usage of GPS for technology designed to improve running. The GPS accuracy levels of any android device can be altered to balance the accuracy and power usage. The following test were undertaken to test the GPS:

Static test

The static test is performed to investigate the noise of the position sensor. The test is done by standing in one position and recording the what the GPS measures as the latitude and longitude over time. Figure 4.1 shows the results of this test for 25 seconds.

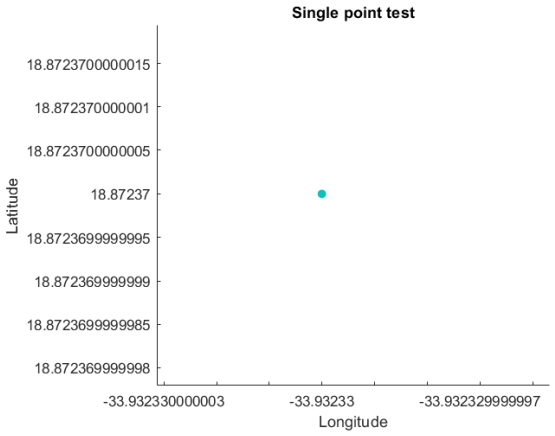


Figure 4.1: Single point accuracy test results

Based on the results on Figure 4.1, the GPS looks accurate when the object is static. The recorded latitude and longitude data is accurate to at least 4 decimal places as can be seen on Figure E.9b. One degree of change in latitude or longitude is equivalent to 111 km of distance. This means there is a 11.1 m confidence on the recorded latitude and longitude points measured. This means that the maximum error from a single measured point is $15.7m$. This occurs when both the measured latitude and longitude points are 11.1 m away from the actual current position. This analysis reveals than even with a 4 decimal place accuracy, the expected accuracy is too low and this requires mitigation. $15.7m.s^{-1}$ is not the typical long distance running speed hence runners would struggle to keep up with a pacer that requires them to move at $15.7m.s^{-1}$. The mitigation method is discussed in subsection 4.2.2.

Altitude test

The altitude test is performed to investigate the accuracy of the measured altitude. Similar to the static test, multiple altitude recordings are made over time. The object stays at the same elevation to track the accuracy of the measured altitude level. Figure 4.2 shows the results this test.

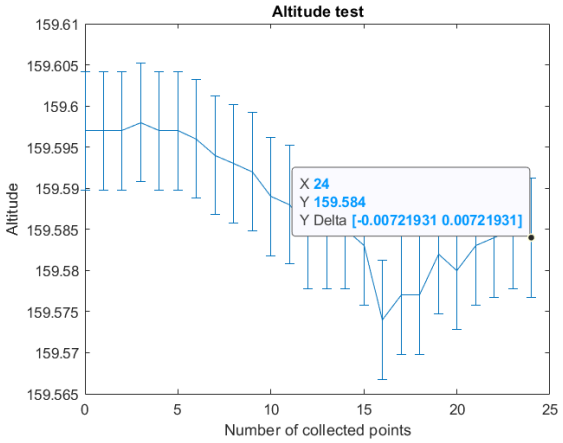


Figure 4.2: Altitude accuracy test results

It is clear that the accuracy level of the measured altitude is not a hundred percent as expected. The standard deviation of the measured results is 0.00721931 m. Considering the use of the altitude data on the project, as will be seen in the next section, the deviation in the measured altitude level is small enough to be negligible.

Dynamic test

The running test is performed to check the repeatability of the recorded data. To investigate the variance on route data while running for repeated recordings of the same route. This is done further understand the expected error margin of the recorded results.

4.2. Software

4.2.1. Matlab Mobile

This subsection details the steps taken to collect route data through Matlab mobile and import them to Matlab as the input to the autonomous pacer algorithm.

Step 1

Firstly, Matlab mobile is downloaded from google play store onto the host device. This process is shown in subfigure 4.3a.

Step 2

Once Matlab mobile is downloaded and installed onto the host device, a mobiledev object [10] is created to read sensor data from the host device. This is done on the command prompt of Matlab mobile as shown in subfigure 4.3b. If data is to be streamed to Matlab, the "logging" variable of the created mobiledev object must be set to 1.

Step 3

Once step 2 is competed, the sensors of the host device can be accessed as shown in subfigure 4.3c.

Step 4

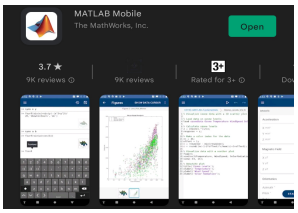
For this application, the sensor is set to stream the recorded data to the same Matlab workspace Matlab mobile was working on at 1 Hz as shown in subfigure 4.3d. The sensors can also be set to log and save the recorded data onto the host device and the data can be manually imported onto the Matlab workspace.

Step 5

When the setup is complete, the person collecting the data must go to the starting point of the intended running course and transverse the whole course. Position data shown in subfigure 4.3e will be recorded every second as specified by the sample rate in subsection 4.2.1.

Step 6

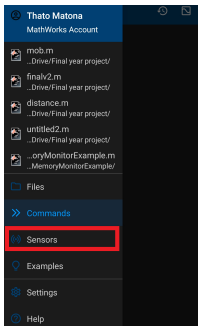
When the finish point of the course is reached, streaming must be stopped as shown in subfigure 4.3f, and the data saved onto Matlab.



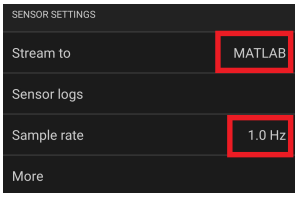
(a) Downloading Matlab Mobile

```
>> m = mobiledev
```

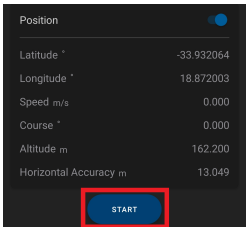
(b) Creating a mobiledev object



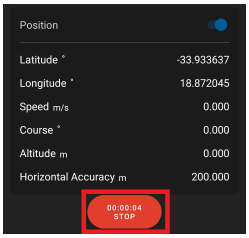
(c) Accessing sensor functions



(d) Setting up the streaming environment and sample rate



(e) Recording the route data



(f) Saving the recorded route data

Figure 4.3: Illustration of the steps

4.2.2. Clean data

The data collected by the android device is not perfectly accurate. The pacer algorithm will rely on resolution of the data for consistency in the pacing updates. This was done by increasing the resolution of the recorded course data and eliminating points which indicate a stop during the recording of the data. This means that there is no need for the person recording the course data to move at very slow speeds to try and get a high resolution course. This approach also reduces the minimum 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 as they require. However, if 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 [11], the distance between two latitude and longitude sets can be calculated using the haversine formula as follows:

The haversine formula shown in equation 4.1 is translated to include latitude, ϕ, and

longitude, ψ, coordinates as shown in equation 4.2.

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

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

$$c = 2 * atan2(\sqrt{a}, \sqrt{1-a}) \tag{4.3}$$

Knowing the mean radius of the earth, R to be 6 371 km, the distance between the two coordinates can be found as shown in equation 4.4

$$d = R * c \tag{4.4}$$

Using equations 4.2, 4.3 and 4.4, the distance between successive recorded latitude and longitude was found. This is shown in figure E.1.

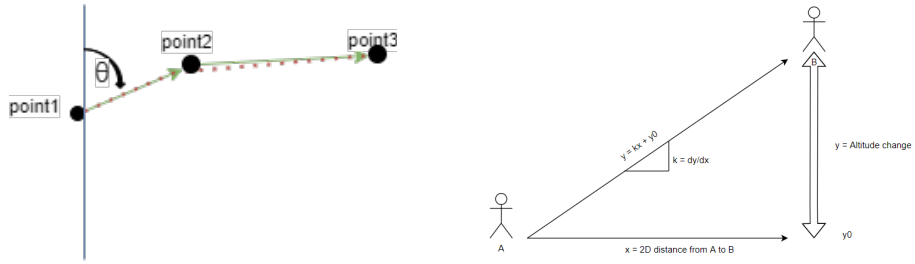
The resolution of the course data can be increased by decreasing the distance between subsequent data points. The resolution can be controlled and was chosen as 0.2m to balance simulation time and accuracy. The change in resolution was done by finding the first set of coordinates and incrementing their position by the chosen resolution value with regard to the current bearing and to find the coordinates 0.2 m away in the right direction. This is done until the total incremental from two subsequent recorded data sets is more then the calculated distance between them. 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. Subfigure 4.4a shows new equally spaced coordinate sets represented by the red dots. These sets are found from extrapolating 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, $cos(x) \approx 1$, 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 [12] were used to find the coordinates at the incremental distance. These equations are shown in 4.5 and 4.6 with δ being the travelled angular distance and θ the bearing, clockwise from true north. Equation 4.5 finds the next latitude point and equation 4.6 finds the next longitude point.

$$\phi_2 = \phi_1 + sin^{-1}(sin\phi_1 * cos\delta + cos\phi_1 * sin\delta * cos\theta) \tag{4.5}$$

$$\psi_2 = \psi_1 + atan2(sin\theta * sin\delta * cos\phi_1, cos\delta - sin\phi_1 * sin\phi_2) \tag{4.6}$$

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 of the linear change becomes the altitude change between the subsequent recorded coordinate sets divided by the distance between the points. This is shown below in the subfigure 4.4b.



(a) Extrapolating new coordinate sets from the recorded sets

(b) Linearising altitude change for subsequent coordinate sets

Figure 4.4: Extrapolating new coordinates and their respective approximate altitudes

For a set resolution of 0.2 m, as shown in Figure E.2, the resulting distance between subsequent coordinate sets is shown in Figure E.3. The resolution of the course can be controlled.

4.2.3. Speed and distance calculator

Using equations 4.2, 4.3 and 4.4 from subsection 4.2.2 the course distance, d_{total}, is recalculated. Once the user inputs the desired completion time, $t_{completion}$, the average speed is calculated as follows:

$$v_{average} = \frac{d_{total}}{t_{completion}} \tag{4.7}$$

The course is divided into segments of equal distances. The division is done by assigning a value and grouping the segments to have the assigned value of coordinate sets. The segments will have equal distances since all the points are equidistant from each other.

4.2.4. Controller

As stated in the scope in section 1.6, the effects of weather will not be focused on therefore by neglecting drag and assuming wind speed to be zero, equation 2.5 simplifies to,

$$P = cmv + imgv \tag{4.8}$$

To eliminate the dependence on mass, specific power,

$$P_s = cv + igv, \tag{4.9}$$

is used over power. It can be observed that ECOR is directly proportional to the specific running resistance, $P_{sr} = cv$. 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,

$$P_s = P_{sr} = cv \tag{4.10}$$

When a gradient is introduced on a course, to keep the ECOR constant, the speed must change.

$$cv = cv_{new} + igv_{new} \tag{4.11}$$

The new speed is,

$$v_{new} = \frac{v}{1 + \frac{ig}{c}} \tag{4.12}$$

If ECOR is constant, knowing the gravitational acceleration is constant, the only factor that can affect the change in speed is the change in gradient ascent or descent. When the gradient increases, the new speed will be less than the old speed and conversely. This is shown in figure E.5. The energy and specific power is keep constant resulting is more optimal running than just controlling the power output of the runner. The results are scaled such that the completion time is as specified by the user. This process is shown in equation 4.13. 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.

$$v_{new} = v_{new} * (\frac{v_{oldaverage}}{v_{newaverage}}) \tag{4.13}$$

4.2.5. Visualiser

As mentioned in subsection 3.2.5, the geoplayer function was chosen as a way to visualise the results. The geoplayer was set as shown in figure E.6. Dummy runner data is fed to the visualiser and compared with the data from the pacer algorithm (Clean data in subsection 3.2.2, Speed and distance calculator in subsection 3.2.3] and Controller in subsection 3.2.4). This is shown in figure E.7. A sampling time of one second was chosen as is the typical sampling time of most fitness apps. Figure 4.5 shows the output of the visualiser.

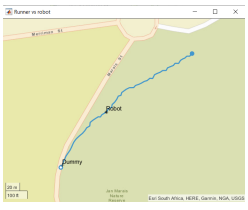


Figure 4.5: Visualiser output

When the runner is behind Robot, the message, "You are behind" is displayed. When the Robot 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 robot's position. If they follow the indicators, they will finish the run in the most optimal running fashion.

Chapter 5

Practical Implementation

With the autonomous running pacer being developed and tested, this chapter will discuss and illustrate how it can be practically implemented for runners across the world to use in their quests for personal excellence.

5.1. System Overview

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

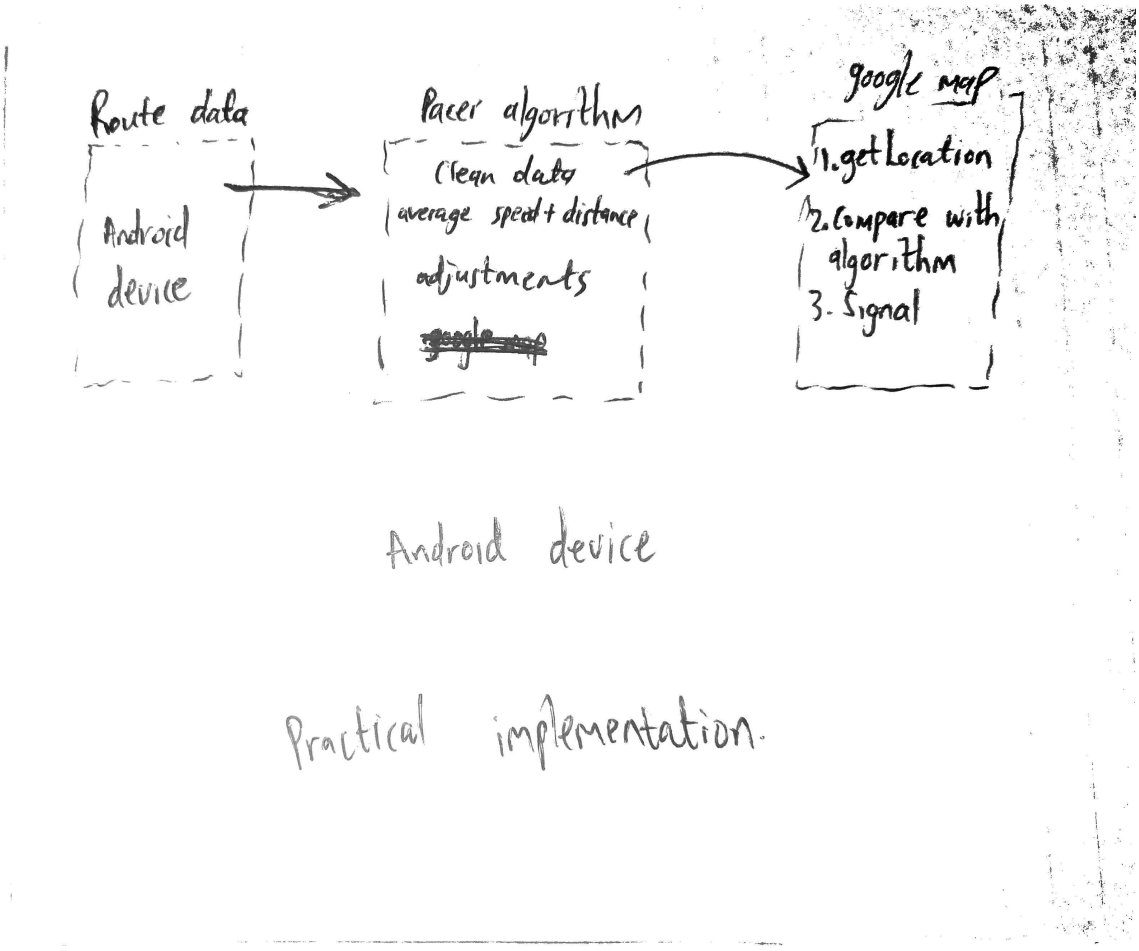


Figure 5.1: System overview on a host device

Route data is collected through the runner's android device and saved onto the host application. The algorithm will use that route data and set 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 algorithm and a 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.

5.2. Implementation

There are a number of ways the algorithm could be implemented. One that have been identified using the matlab andriod support packages and let the algorithm run through matlab mobile. Another is integrating the algorithm with existing fitness applications such as Strava, Zepp Life, NRC, Garmin Connect, Zombies Run and many more. The latter approach is investigated in this section of the report. The advantage of the latter approach is accessibility. As stated on the android studio IDE and on the android police website [13], the matlab android support package compiles for minimum API 30 android devices and only 24.3% of android devices have an API 30 SDK or above. Runners are already using the popular fitness apps so it makes sense to improve their running experience and not change it. Also, all of the mentioned fitness applications already collect the data required by the algorithm to work. This would make the integration process close to seamless. For the purpose of this report, a standalone android application was created to illustrate how the algorithm could be implemented for usage. The following sections briefly discuss how the implementation process.

5.2.1. Acquiring Route Data For The Host App

The challenge of acquiring usable running route data was discussed in chapter 3, hence will not be discussed again here. To demonstrate usability of the pacer, route data will be acquired exactly the same way as in previous stages of the project. This is done only to not need to rewrite matlab code as java or kotlin code for the same functionality as the matlab code. None of the code used is matlab specific, see appendix **??**. This saves time while still demonstrating the usability of the autonomous pacer. Given that potential host apps for the algorithm are popular fitness applications such as Strava ,an assumption that the host app is able to acquire the data needed by the algorithm is made.

5.2.2. Producing the autonomous pacer output

It is important to note that the Robot.txt file shown in figure 5.3 would be produced by the host application. Figure 5.2 (MAYBE PUT ON APPENDIX) show how the autonomous

pacer output is produced.

```
11      fileID = fopen('Robot.txt','w');
12      fprintf(fileID,'%3.7f,%3.7f\n',[results(:,1) results(:,2)]);
13      fclose(fileID);
14
```

Command Window

```
-33.9320764,18.8719731
-33.9320852,18.8719546
-33.9320854,18.871954
-33.9320797,18.871971
-33.9320839,18.8719608
-33.9320812,18.8719614
-33.9320809,18.8719618
-33.9320826,18.8719617
-33.932083,18.8719616
-33.9320831,18.8719615
-33.9320831,18.8719614
-33.9320769,18.8719746
-33.9320765,18.8719752
-33.9320766,18.8719752
-33.9320767,18.8719749
-33.9320837,18.871958
-33.9320849,18.8719553
```

Figure 5.2: Pacer algorithm output

5.2.3. Importing the autonomous pacer output

Again for the purpose of this report, the autonomous pacer output - Robot.txt - is imported from Matlab into the host device's memory as shown in figure 5.3.

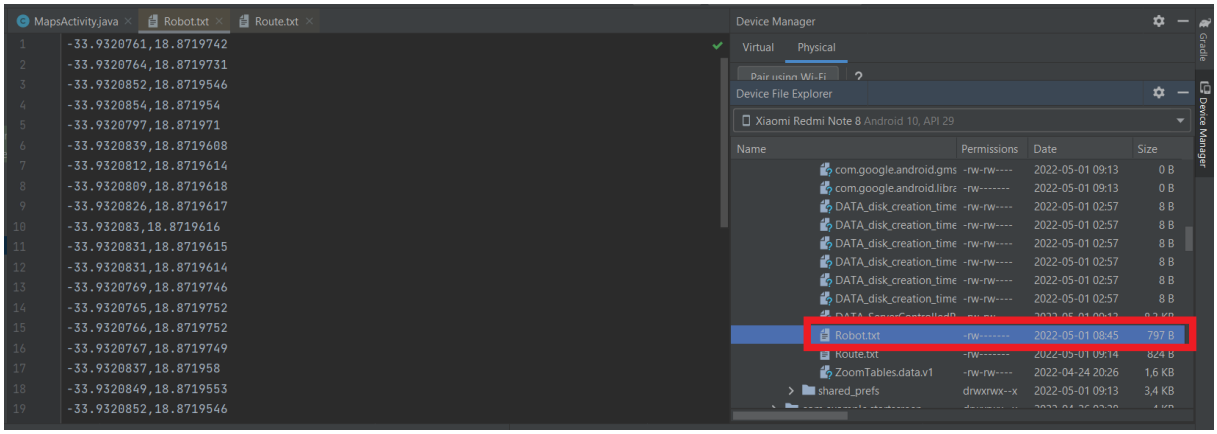


Figure 5.3: Importing the pacer algorithm's output into host device

5.2.4. Comparison and indicators

When the program is started, the live location of the runner is updated every second and compared to the output of the pacer. When the runner is more than a step interval behind the pacer, an indication that they need to speed up is given. When the runner is more than step interval ahead of the pacer, an indication that the runner must slow down is

given. When the runner is within a step interval of the pacer, an indication that they are on pace is given.

5.3. Product Transition

Conclude the workings of the pacer on as an application or integrated application piece. Show example (finishing time, follow pacer, show change in elevation, show speed) The project demonstrated how the pacer would be implemented on an android device but could be used in a similar fashion for all fitness devices including IOS devices. The would only be a difference in the platform of development but the idea and algorithm remains the same.

Chapter 6

Experiments and Results

This chapter discusses the target metrics, the experiment design procedure to test those target metrics and illustrate the results of those experiments. And finally, it discusses the meaning of the results in relation to the project objectives.

6.1. Target Metrics

The pacer's **completion time** is the first metric of interest. The pacer is supposed to enable a runner to finish the given running course at a set time. The completion time of the pacer 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 **altitude** change is also measured to observe how the pacer adjusts the proposed speeds for changes altitude. And finally, the **ECOR** is tested to investigate if the pacer truly allows for optimal running.

6.2. Experiment Design Procedure

In section 1.4, objective 4 and 5 project state that the pacer must be able to adjust paces with respect to the elevation throughout the run and provide a location profile for the runner respectively. This section will test how effective the pacer is in those two regards and in effect test the effectiveness of the developed pacer. This will be done by comparing the pacer's completion time with the desired completion time. Then checking the speeds at different segments relative to the gradient on those segments. Then also by getting the output location profile of the pacer to see if it is attainable. The other metric that will be tested is the ECOR. An assumption for initial power will be made as the power is not measured by the simulation set up. The above will illustrate the effectiveness of the pacer.

6.3. Experiment Results

Different experiments were set to test the behaviour of the pacer for different conditions. Subfigure 6.1a shows how the course data was loaded onto matlab and how the desired completion time was set.

```
load('set12el.mat'); % loads matlab sensor data
oldlat = Position.latitude;
oldlong = Position.longitude;
alt = Position.altitude;
course = Position.course;
size = length(oldlat);
fpc = 15; % frequency of pace change.
x = 180; % completion time in x seconds. Modified by user
```

(a) Loading course data and setting the desired completion time

```
completionTime =

   180
```

(b) Completion time

The simulation was ran and the completion time was 3 minutes as expected. This is shown in subfigure 6.1b. Figure 6.2 shows the location profile of the robot on a map. The changes in the spacing of subsequent coordinate sets is a reflection of the change in speed at different segments. The higher the gradient - as can be seen on subfigure 6.4b - on a segment, the smaller the change in location per update.

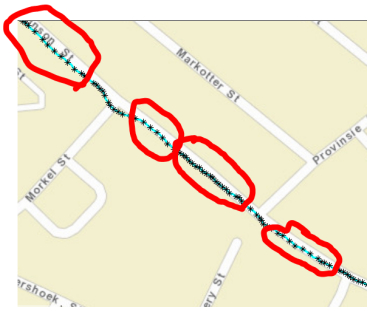
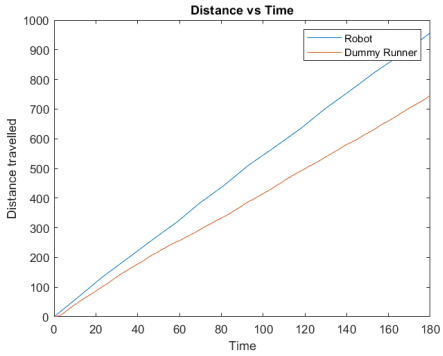


Figure 6.2: Robot location profile on map

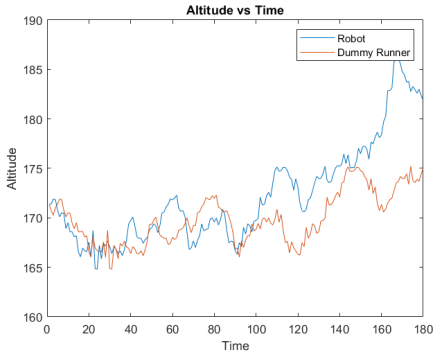
The pacer responds to the runner's deviation from proposed paces by alerting them every second. Figure 6.3 shows what color will be seen in the location profile in relation to the distance between the runner and the pacer.

Figure 6.3: Legend

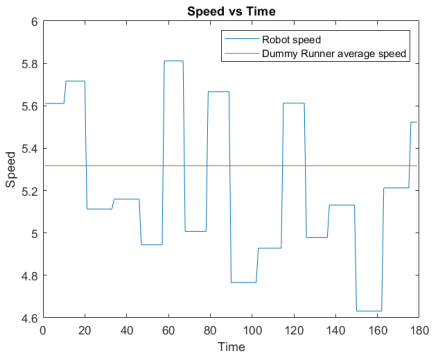
6.3.1. Experiment 1: Runner slower than pacer on a course with altitude changes



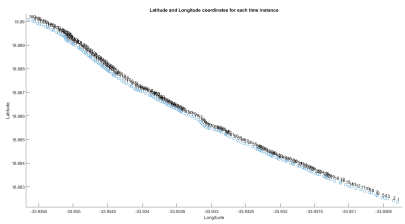
(a) Distance ran by Robot and Runner vs Time



(b) Robot and runner's altitude



(c) Corresponding Robot Speed

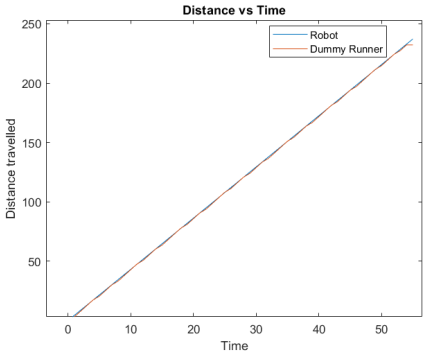


(d) Robot location profile

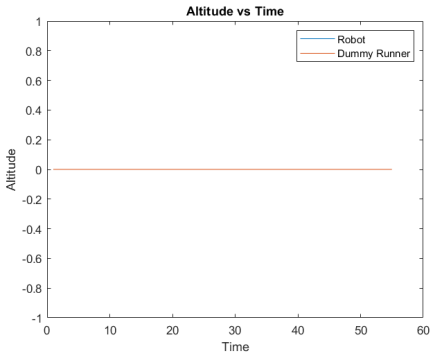
Figure 6.4: Effects of altitude changes on the resulting speed and location profiles

6.3.2. Experiment 2: Runner correctly following pacer on flat course

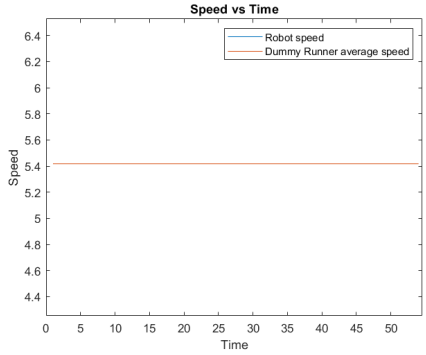
A different course was loaded to illustrate how the algorithm behaves on a flat course. The results are shown on figure 6.5. The desired completion time for this course was set to be 55 seconds.



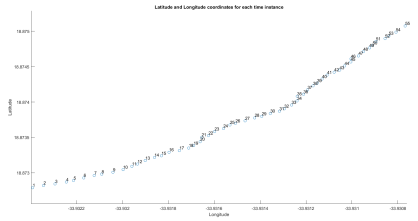
(a) Distance ran by Robot and Runner vs Time



(b) Robot and runner's altitude



(c) Corresponding Robot Speed

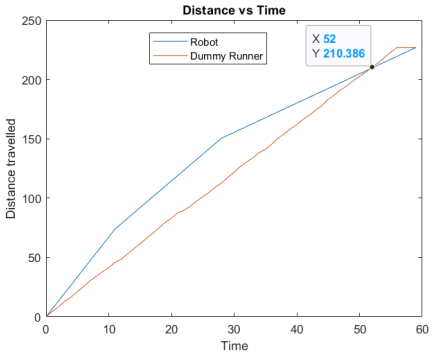


(d) Robot location profile

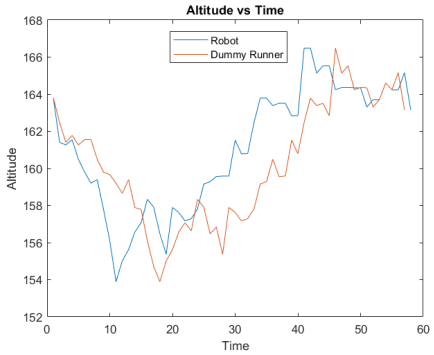
Figure 6.5: Altitude and speed results

For a flat running course, the robot's speed remains constant throughout as expected.

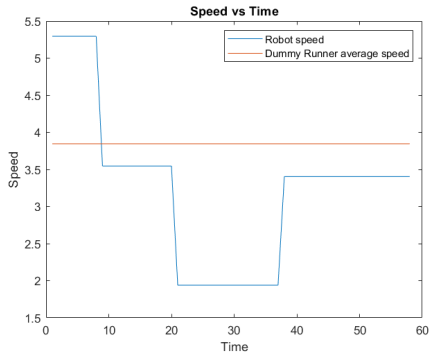
6.3.3. Experiment 3: Runner trying to following pacer on a course with altitude changes



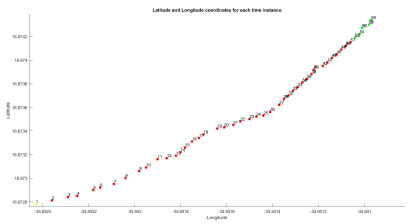
(a) Distance ran by Robot and Runner vs Time



(b) Robot and runner's altitude



(c) Corresponding Robot Speed



(d) Robot location profile

Figure 6.6: Altitude and speed results

6.4. Discussion

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.

Bibliography

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

[2] 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.

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

[4] ——, "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.

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

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

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

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

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

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

[11] 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.

[12] 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.

[13] 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

getGnattChart. In final year project folder.

Appendix B

ECSA Outcomes Compliance

This is another appendix.

B.1. Problem solving

Demonstrated in chapter 1.3 and throughout the project. Elaborate. I took an ill defined problem, understood what I needed to do to solve it, then solved it.

B.2. Application of scientific and engineering knowledge

A scientific/systematic approach - questioning, hypothesis, experimentation, result analysis, conclusion and repetition - will be used to solve the given problem and subproblems encountered. Engineering knowledge and technology such as sensor electronics, robotics, automation, design, existing algorithms, simulations, machine learning, imagine processing etc will be used.

B.3. Engineering design

Design of trajectory planning, tracking, guidance, speed control and collision avoidance systems will be done, applying a number of learnt and new engineering concepts.

B.4. Investigation, experiments and data analysis

Research will be undertaken in the development stage to learn more about current solutions and to determine the best design approaches and optimization alternatives. Experimentation, testing and data analysis will be performed during the design and implementation stages to ensure the running pacer works as intended. Various possible improvements will be considered and implemented after result collection and analysis. Possible future recommendations and integration suggestions will be made depending on the performance of the final design.

B.5. Engineering methods, skills and tools

Throughout the project, skills within and extending out of engineering will be practiced. Time management, planning, communication, design, experimentation, estimation, result interpretation and more. Engineering methods learnt in mathematics, control systems, electronics, computer programming and computer science will be used. Sustainability considerations will be made. Possible use case scenarios and end of life (integration or end of life) will be considered. Simulation tools such as Matlab will be paramount to the completion of the project.

B.6. Professional and technical communication

To be edited at the end. Encapsulate what happened throughout the project.

B.7. Individual work

To be edited at the end. Encapsulate what happened throughout the project.

B.8. Independent Learning Ability

To be edited at the end. Encapsulate what happened throughout the project. What information did I draw/acquire to solve my particular problem? Typically, stuff not learnt in EandE. Demonstrate that you are able to absorb information, learn even from outside of the degree. I needed to know this before solving problem.

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.

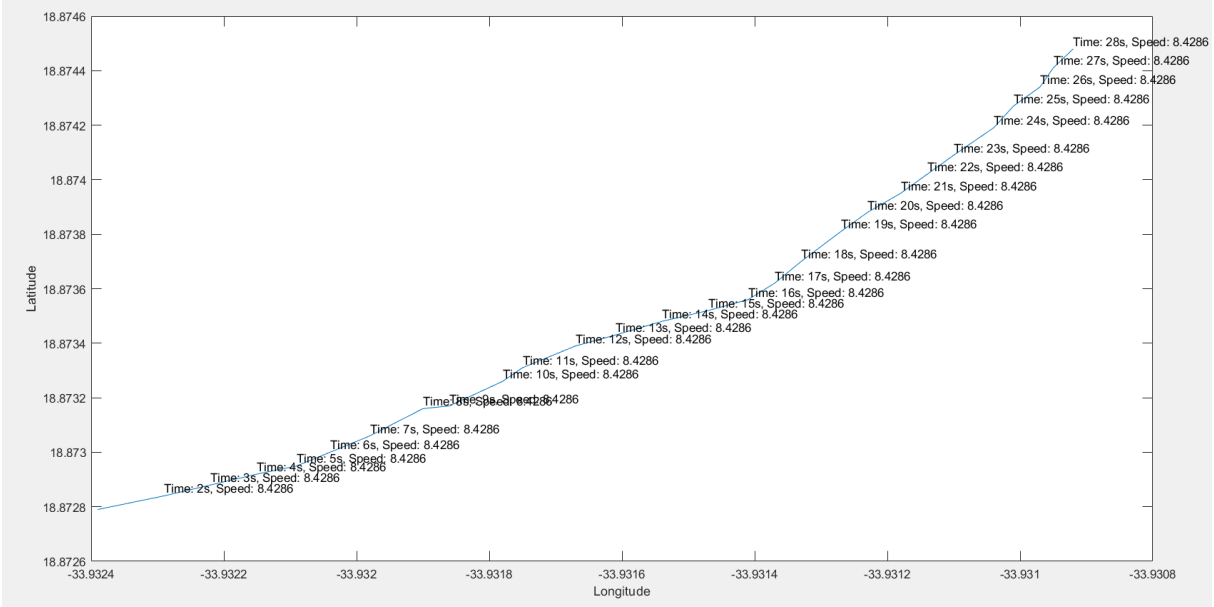
Appendix E

Code Used

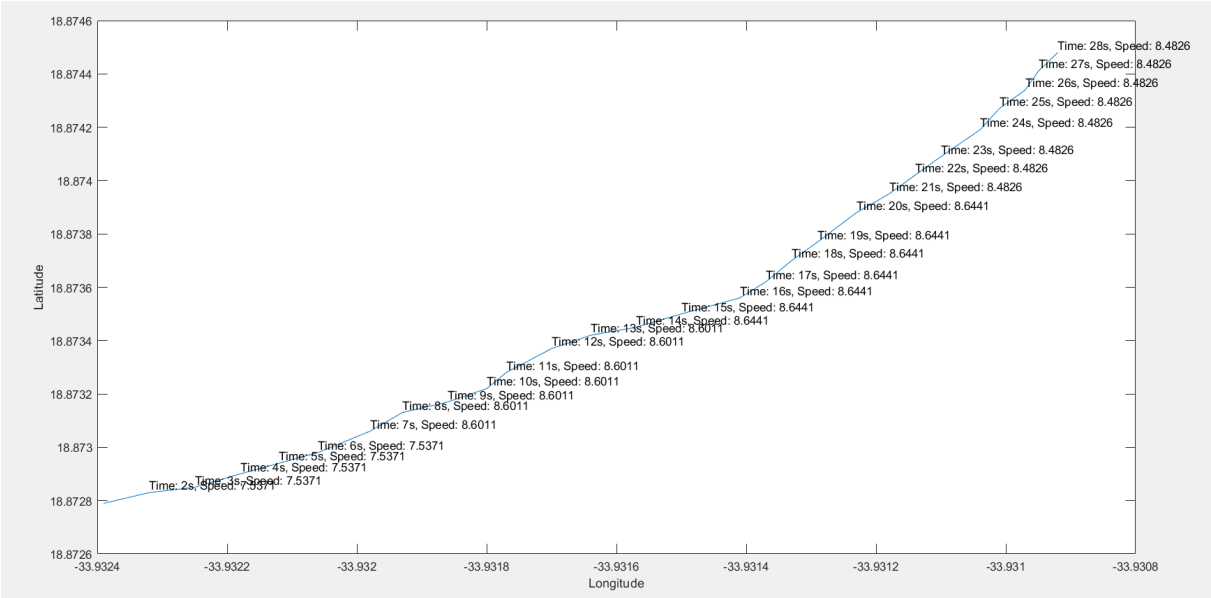
Ghost Pacer YouTube: https://www.youtube.com/watch?v=wKEW$_c$6$NEIU$

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.

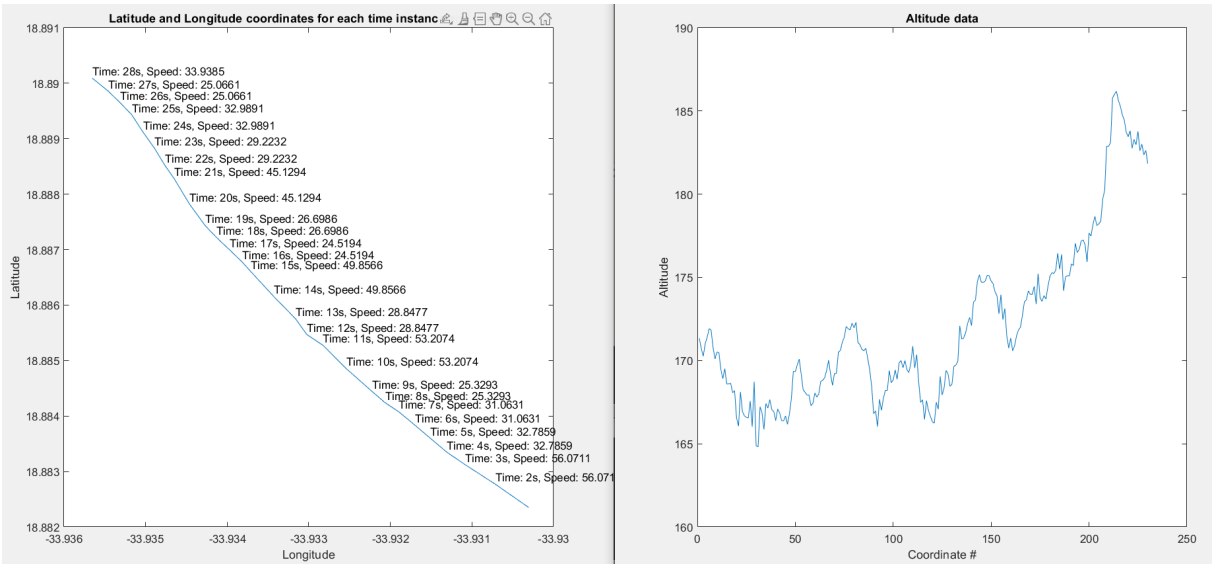


There is a time error introduced by the following line of code,

```
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, NewLatLongAlt, NewSize] = distanceIncr(lat,long,alt,course,size)
    R = 6371000;
    df = zeros(1);
    dInc = [];
    NewLatLongAlt = [lat(1) long(1) alt(1)];
    lat1 = lat(1)*pi/180;
    long1 = long(1)*pi/180;
    d_move = 0.2;

    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;
        dcheck = 0;
        while(dInc(i-1)>dcheck)
            lat2 = lat1 + asin(sin(a1)*cos(d_move/R) + cos(a1)*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));
            NewLatLongAlt = cat(1, NewLatLongAlt, [lat2*180/pi long2*180/pi (alt(i)-alt(i-1))/dInc(i-1)*dcheck+alt(i-1)]);
            long1 = long2;
            lat1 = lat2;
            dcheck = dcheck + d_move;
        end
    end
    NewSize = length(NewLatLongAlt(:,1));
end
```

Figure E.2: Increasing the course resolution

newdInc

1215x1 double

	1
1	0.2000
2	0.2000
3	0.2000
4	0.2000
5	0.2000
6	0.2000
7	0.2000
8	0.2000
9	0.2000
10	0.2000
11	0.2000
12	0.2000
13	0.2000
14	0.2000
15	0.2000

Figure E.3: Resulting distance between subsequent coordinates

```
load('matlabflat.mat'); % loads matlab sensor data
lat = Position.latitude;
long = Position.longitude;
alt = Position.altitude;
course = Position.course;
size = length(lat);
st = rateControl(5);

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

player = geoplayer(lat(1),long(1),18);
player.Parent.Name = 'RecorderHD';
player.Basemap = 'streets';
plotRoute(player,nll(:,1),nll(:,2));

for i = 1:length(nll(:,1))
    plotPosition(player,nll(i,1),nll(i,2),"TrackID",1,"Marker","*","Label","HD");
    distance = newdf(i)
    time = st.TotalElapsedTime
    waitfor(st);
end
```

Figure E.4: Clean data testing

```
function newspeed = segmentTimes(avgspeed, alt, fpc, dps, tdt)
    size = length(alt);
    elevation = [];
    step = 1;
    for i = 1:1:size+1
      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

    newspeed = [];
    televation = sum(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
        newspeed = cat(1,newspeed,newspeed(length(newspeed)));
    end
end
```

Figure E.5: Calculating new speeds for each segment

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

Figure E.6: Visualiser setup

The dummy data in figure E.7 is the runner at the speed while the data was recorded and the robot data is the output of the controlled speed.

```
while(i<=size)
    if i<=size
        plotPosition(player,lat(i),long(i),"TrackID",2,"Marker","+","Label","Dummy");
    end

    if i<=x-1
        if dist(i) <= df(k)
            i = i + 1;
            results = cat(1,results, [lat(k) long(k)]); % coordinates at this time instance
            plotPosition(player,lat(k),long(k),"TrackID",1,"Marker","*","Label","Robot");

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

            speed{i} = speed_per_segment(step);
            current_speed = speed_per_segment(step)
            distanceRan = dist(i)
            TimeElasped = sampleTime.TotalElapsedTime
            if k>i
                'You are behind'
            end
            waitfor(sampleTime); % wait for 1 second
        else
            k = k + 1;
        end

    else
        i = i + 1;
        plotPosition(player,lat(size),long(size),"TrackID",1,"Marker","*","Label","RobotDone");
        waitfor(sampleTime);
    end

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

Figure E.7: 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.8: Hardware testing

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

Position

25x6 timetable

	Timestamp	1 latitude	2 longitude	3 altitude
1	15-May-2022 08:44:44.000	-33.9323	18.8724	159.5970
2	15-May-2022 08:44:45.000	-33.9323	18.8724	159.5970
3	15-May-2022 08:44:46.000	-33.9323	18.8724	159.5970
4	15-May-2022 08:44:47.000	-33.9323	18.8724	159.5980
5	15-May-2022 08:44:48.000	-33.9323	18.8724	159.5970
6	15-May-2022 08:44:49.000	-33.9323	18.8724	159.5970
7	15-May-2022 08:44:50.000	-33.9323	18.8724	159.5960
8	15-May-2022 08:44:51.000	-33.9323	18.8724	159.5940
9	15-May-2022 08:44:52.000	-33.9323	18.8724	159.5930
10	15-May-2022 08:44:53.000	-33.9323	18.8724	159.5920
11	15-May-2022 08:44:54.000	-33.9323	18.8724	159.5890
12	15-May-2022 08:44:55.000	-33.9323	18.8724	159.5880
13	15-May-2022 08:44:55.999	-33.9323	18.8724	159.5850
14	15-May-2022 08:44:57.000	-33.9323	18.8724	159.5850
15	15-May-2022 08:44:58.000	-33.9323	18.8724	159.5850

(b) Hardware test set