Aleksander Avastu

SIMULATING CONTROL DATA FOR A REAL-TIME TRACKING ALGORITHM

Bachelor’s Thesis (9 ECTS)

Supervisor: Artjom Lind, MSc
Simulating control data for a real-time tracking algorithm

Abstract:
Due to the ease of access to increasingly complex technology, the expectations of such consumers have risen as well. In order to meet the expectations of the consumers, advances are being made in the field of computer science on a daily basis. However, a major challenge for real-time tracking algorithms is testing or more accurately, verifying the accuracy of an algorithm. The most common way of overcoming this hurdle is testing the algorithm in real life. This is done by doing accurate measurements and comparing them with the predicted data of the algorithm. The biggest problem with this approach is the potential cost of the testing. Such tests often require delicate machinery being operated by professionals.

An alternative way of testing real-time tracking algorithms is by creating a simulation, which represents real life situations and comparing the simulated data with the predicted data of the algorithm. By creating a simulation for testing purposes, scientists can avoid buying or renting expensive equipment and still get valuable information, needed for their algorithms. Another factor in favour of simulations is the possibility of near endless accuracy. With simulations it is completely feasible to gather state data a thousand, a million or even a billion times a second.

Due to the requirements set for the selected algorithm, this thesis focuses on providing suitable state data of the simulated vehicle once every millisecond, for ten seconds total.

Keywords:
Real-time tracking; Simulating control data; Tracking algorithm

CERCS:
P175 Informatics, systems theory
Kontrollandmete simuleerimine reaalajas asukohta jälgimise algoritmi jaoks

Lühikokkuvõte


Võtmesõnad:
Reaalajas asukohta jälgimine; Kontrollandmete simuleerimine; Asukohta jälgimise algoritm

CERCS:
P175 Informaatika, süsteemiteooria
# Table of contents

Acknowledgements ........................................................................................................... 6

Abbreviations and Acronyms .......................................................................................... 7

1 Introduction .................................................................................................................... 8

1.1 Motivation .................................................................................................................. 8

1.2 Objectives .................................................................................................................. 8

1.3 Limitations and challenges ....................................................................................... 9

1.4 Road map ................................................................................................................... 9

2 State of the art ............................................................................................................. 10

2.1 Unity .......................................................................................................................... 10

2.2 Unreal engine ............................................................................................................. 11

2.3 GameMaker .............................................................................................................. 11

3 Methodology ................................................................................................................ 12

3.1 The tracking algorithm ............................................................................................ 12

3.2 Technologies and programming languages used .................................................... 13

3.2.1 Python .................................................................................................................. 13

3.2.2 Node.js ............................................................................................................... 13

3.2.3 TypeScript .......................................................................................................... 14

3.2.4 Node Package Manager ..................................................................................... 14

4 Implementation .......................................................................................................... 15

4.1 The prototype .......................................................................................................... 15

4.2 Final iteration .......................................................................................................... 15

5 Analysis ....................................................................................................................... 18

5.1 Result of the simulation ......................................................................................... 18

6 Conclusion and future research .................................................................................. 20

References ...................................................................................................................... 21

Appendix ......................................................................................................................... 23
I. Usage .................................................................................................................. 23

II. Licence ............................................................................................................. 24
Acknowledgements

First and foremost I would like to thank my supervisor Artjom Lind for helping me with well, everything. He was the one who helped me settle on such an interesting topic. I especially appreciate his way of guidance, always keeping a watchful eye on the progress of the thesis whilst still giving me complete freedom over the way it was written.

Secondly I would like to thank my friends and family for helping me find motivation when little was to found.
Abbreviations and Acronyms

This section gives an overview of abbreviations and acronyms used throughout the paper.

GPS – Global Positioning System
2D – Two-dimensional space
3D – Three-dimensional space
VR – Virtual reality
AR – Augmented reality
API – Application programming interface
UE – Unreal engine
UE4 – Unreal engine 4
FAQ – Frequently asked questions
I/O – Input/Output
CLI – Command-Line Interface
GUI – Graphical User Interface
1 Introduction
With the ever growing popularity of integrating smartphones into people’s everyday lives, the need for seamlessly operating software has risen drastically. This is also true in the field of real-time location tracking. Smartphones are fitting enablers for such a need. Almost all smartphones today are equipped with a built-in accelerometer and a gyroscope. This allows computer scientists to provide the end users with a more intuitive and accurate user experience. In order to provide the end users with the best experience possible, a multitude of algorithms for real-time location tracking have been developed [1, 2]. However there are certain difficulties when testing such algorithms, such as finding or collecting control data. Here, in this thesis we aim to show that the process of collecting real-time data from a sufficiently accurate device can be overstepped by simulating the conditions of a moving vehicle.

1.1 Motivation
The need for this thesis originates from the issue of accessibility to control data for real-time tracking algorithms. This is caused by either a lack of access to testing equipment due to financial reasons or algorithm developers withholding test data in order to preserve a competitive edge. Thus, another solution for validating the accuracy of tracking algorithms is needed.

By simulating a vehicle’s movement, the author can produce a controlled dataset which could provide algorithm developers with a dataset for an accurate testing process. With ease of access to control data not only would more people be inclined to opt for simulated data, developers of other applications which take advantage of the smartphone’s built in gyroscope and accelerometer could test their applications in more robust conditions whilst still being in a controlled environment.

1.2 Objectives
The main objective of this thesis is to create an open-source, cross-platform simulator of a moving vehicle. In order to achieve this goal, the following sub-objectives have been defined:

- Determine necessary parameters and their required formats for the simulation
- Research existing simulators of moving objects
- Determine required formulas for simulating a moving object
- Implement simulation formulas
• Export dataset of state data for each millisecond of the vehicle

1.3 Limitations and challenges
The greatest hurdle was the lack of any existing simulators which would fit the criteria. This meant that it was necessary to develop the acceleration, deceleration, altitude change and the direction change functionality from scratch, whilst still supporting the gyroscope and accelerometer expected data outputs.

1.4 Road map
The dissertation of the thesis has been structured in the following order.

Chapter 2: Gives insights into the current state of the art in regards to simulation creation for real-time tracking algorithms.

Chapter 3: Introduces different technologies used in the creation of the simulation whilst providing the reader with an explanation of why the technologies were chosen.

Chapter 4: Explanation of the simulation. Provides the reader with an overview of the formulas used.

Chapter 5: Introduces the reader to the results of the simulation. Gives an overview of points of interest.

Chapter 6: Concludes the thesis and gives additional ideas for improving the simulation.
2 State of the art

The aim of this chapter is to provide the reader with an overview of technologies and techniques used in the creation of the real-time tracking simulation. In addition, this chapter aims to give additional insights into real-time tracking algorithms with the main focus being on algorithms which use multi-sensor fusion in addition to only using GPS coordinates.

A common method of developing control data for a GPS is using game engines. Most game engines are shipped with coordinate systems out-of-the-box. This makes simulating moving vehicles simplistic. The biggest drawback of game engines in the scope of this thesis is the lack of support for simulating gyroscope data or accelerometer data. Because of this, the simulation of the gyroscope sensor or the accelerometer sensor would have to be written from the ground up.

2.1 Unity

Unity is a game engine which is used to create 2D, 3D, VR and AR experiences. Though the main use of the engine was initially for creating games, over time the usage of the engine has grown to include areas outside the gaming industry [3].

In 2019, Unity Technologies announced their newest product, Unity Simulation [4]. Unity Simulation promises to bring the power of Unity to the cloud. Unity Simulation will allow researchers perform large and complex simulations on a scalable platform. Due to the Unity Simulation still being in beta at the time of writing, as well as the Unity free license [5] restrictions, the usage of Unity Simulation for generating control data for a multi-sensor fusion algorithm is not viable.

Unity is capable of simulating the motion of vehicles with ease because Unity provides developers with the necessary API’s. One such API is the LocationService API [10], which gives developers and researchers an easy access to the coordinate system of the engine. Such simulations are made even easier with additional assets from the asset store e.g. the Vehicle Physics Pro [6]. While Unity supports gyroscope [7] and accelerometer [8] input data from mobile devices, the greatest shortcoming of Unity in the scope of this thesis is the lack of support for gyroscope and accelerometer data simulation. The lack of support for simulation and the restrictions of the free Unity license mean that Unity can not be used for simulating the necessary data.
2.2 Unreal engine

Similar to Unity, Unreal engine is a game engine used primarily for the creation of video games. UE4, the latest stable version of the game engine uses PhysX3.3 physics engine [9] in order to provide developers and researchers with a lifelike experience. Similar to Unity, UE4 provides an ease of access to its coordinate system with the built in API’s [11]. Automated Driving Toolbox [12] et al. give researchers an alternative to developing their own simulations. The problem in the scope of this thesis with the aforementioned 3D environment is the lack of support for simulating gyroscope and accelerometer data.

The biggest differences between Unity and UE4 – two of the most common modern game engines – are ease of use and licensing. Due to the nature of UE4 leaving more control over the engine to the developer, developing products with UE4 is generally considered harder. Unreal engine’s free license allows developers to create open source projects with ease [13] but due to the lack of neither gyroscope nor accelerometer support, UE4 is not a viable option for this type of simulation.

2.3 GameMaker

GameMaker is a game engine designed for the development of 2D games. In addition GameMaker allows for cross-platform gameplay and is supported on all major platforms [14]. Gamemaker allows for a fairly seamless creation of simulations by enabling physics [15]. The physics engine is not as complex when compared with Unity or Unreal Engine but it is definitely powerful enough for creating simplistic simulations.

Even though GameMaker is intended for the development of 2D games, the creation of 3D games with this engine is also possible [16]. Though this is commonly discouraged due to the fact that the 3D functionality of GameMaker is limited to the graphics part [16]. This means that the usage of GameMaker for a simulation with a 3D coordinate systemFor this reason and the fact that the engine does not natively support neither gyroscope nor accelerometer simulation means that the engine is not a viable base for this GPS data simulation.
3 Methodology

The aim of this chapter is to provide the reader with a short and concise overview of the algorithm for which the simulation was created. In addition to the algorithm overview, the author also highlights few of the technologies and languages used in the creation of this simulation and gives a high-level explanation of why these technologies were chosen.

3.1 The tracking algorithm

The inspiration for this thesis originates from a specific article [2] called “Real-Time Vehicles Tracking Based on Mobile Multi-Sensor Fusion” authored by Siim Plangi, Amnir Hadachi, Artjom Lind and Abdelaziz Bensrhair. Whilst the algorithm described in the aforementioned research paper uses different constants for the computation of accurate coordinates, it takes in three main parameters which are vital for the prediction process:

1) Accelerometer reading
2) Gyroscope reading
3) GPS readings

The accelerometer data is used for measuring the vehicle’s acceleration and deceleration. The gyroscope data is used for the detection of turns which in turn is used for the correction of heading. The GPS data provides the algorithm with speed, heading and most importantly, coordinates.

The algorithm uses three filters which are responsible for smoothing and fusion of data:

1) Exponential moving average filter
2) Low pass filter
3) Kalman filter

The real-time tracking algorithm uses in addition to the filters mentioned before, six processing modules which are used for different detection of state changes. The processing modules are as follows:

1) Gravity calibration
2) Vehicle acceleration
3) Angular speed
4) Speed fusion
5) Vehicle heading
6) Tracking fusion
3.2 Technologies and programming languages used

The goal of this chapter is to give a short overview of programming languages used in the creation of the simulation.

3.2.1 Python

Python is a high-level programming language which is commonly used for general-purpose programming. Developing prototypes and writing easily understandable code is made simple with Python due to the expressive syntax of the language [17]. According to a 2019 survey [19] 49% of Python developers use it for data analysis, and 22% use it for software prototyping. Python was the ideal choice for prototyping however the shortcoming became apparent with the ever-growing list of formulas used in the simulation. Due to the amount of formulas used, with each one using different units of measure, the need for static run-time type checking became necessary in order to keep the code clean.

Python supports type hinting natively since version 3.5 [20]. However the problem with Python’s type hinting is that the compiler does not enforce run-time type checking. This means that the compiler more or less completely ignores the type hinting, making type hinting only useful for code readability. In order to enforce run-time type hinting an external library would have to be used. One such library is mypy which is capable of enforcing run-time static type checking [18].

3.2.2 Node.js

Node.js is an event-driven JavaScript runtime [21] which conversely to JavaScript executes code on the back-end instead of front-end or the client machine. Being a JavaScript runtime means that there are no blocking processes due to Node.js nigh never performing I/O operations. Node.js is designed without threads but using multiple cores is still possible with the child_process API [22]. Node.js supports almost every operation that regular JavaScript does and provides system API’s which other back-end programming languages have access to like the file system API [23].

Node.js supports a plethora of community made packages with the NPM module. This, in tandem with the author’s background being in web development was the reason that Node.js was chosen as the primary programming language for the creation of the simulation of the real-time tracking algorithm.
3.2.3 TypeScript

TypeScript is a superset of JavaScript which takes advantage of type hinting whilst still compiling into plain JavaScript [24]. TypeScript has its own compiler and as such is capable of introducing many new features for the JavaScript language. One of the main features is static run-time type hinting enforcement. TypeScript also introduces many new features which are lacking in the regular JavaScript runtime like optional chaining, interfaces et al. [25]. TypeScript was chosen for this project primarily for its support for type hinting and use of interfaces.

3.2.4 Node Package Manager

NPM is a package manager for the Node.js JavaScript runtime. NPM consists of a CLI and an online database. The online database holds all of the community published packages which can be used by developers world-wide in development and production environments. The CLI of NPM is used for installing, updating and uploading packages. In the scope of this thesis, NPM was used for the installation of TypeScript and its ease of command merging with the “npm run <command>“ command.
4 Implementation

The aim of this chapter is to give the reader an explanation of formulas and implementations made in the process of creating the control data. The chapter is divided into two main sections: the prototype and the final iteration.

4.1 The prototype

The prototype for the simulation was created using Python. The code simulated ten seconds by creating a for loop of iterators over the range of 10 000. In this range of 10 000, each iteration represented 1/1000 of a second.

During each iteration of the loop, the velocity of the vehicle was calculated by using a formula of linear growth.

\[ v_i = t \times 10 \] (1)

Where \( v_i \) is the velocity and \( t \) is time.

After calculating the velocity, the distance travelled was calculated as follow:

\[ d_i = d_{i-1} + t \times v_i \] (2)

Where \( d_i \) is total distance travelled, \( t \) is time, and \( v_i \) is the velocity.

Due to the prototype only using 1 dimensional coordinates and not including neither the gyroscope nor the accelerometer data, no further calculations were made. However, the inflexibilities of the velocity formula became apparent. As such, the prototype was scrapped and a new project was created.

4.2 Final iteration

Node.js in tandem with TypeScript was chosen for the creation of the second and final iteration of the simulation. This iteration borrows the concept of simulating the ten seconds from the prototype, by creating a for loop over a range of 10 000.

For the first three seconds of the simulation, a force is defined which is to be applied to the vehicle. Rolling resistance and air resistance are then subtracted from the total force.
The formula used for calculating rolling resistance is as follows:

\[ F_r = c \times m \times g \] (3)

Where \( F_r \) is the rolling resistance, \( c \) is the rolling coefficient, \( m \) is the mass of the object and \( a_g \) is the acceleration of gravity. The rolling coefficient for air filled tires can be calculated as follows:

\[ c = 0.005 + \frac{1}{p} \times \left( 0.01 + 0.0095 \times \left( \frac{v_i}{100} \right)^2 \right) \] (4)

Where 0.005 is the rolling resistance coefficient constant of air filled tires of asphalt, \( p \) is the tire pressure and \( v_i \) is the velocity.

After calculating the rolling resistance, the air resistance was calculated with the following formula:

\[ F_d = c_d \times \frac{1}{2} \times p \times v^2 \times A \] (5)

Where \( F_d \) is the drag force, \( c_d \) is the drag coefficient, \( p \) is the density, \( v \) is the velocity and \( A \) is the frontal area of the body.

Whilst moving on an incline parallel to the earth, the acceleration of the vehicle is calculated with the following formula:

\[ a_i = \frac{F}{m} \] (6)

Where \( a_i \) is the acceleration, \( F \) is the applied force and \( m \) is the mass.

Whilst moving on a rising or a falling incline, the acceleration of the vehicle is calculated with the following formula:

\[ a = \frac{F \times \cos(\alpha) - m \times g \times \sin(\alpha)}{m} \] (7)

Where \( a \) is the acceleration, \( F \) is the force applied, \( \alpha \) is the angle of the incline, \( m \) is the mass of the vehicle and \( g \) is the gravitational constant. It is visible that formulas 6 and 7 are actually
both mutations of the same law, Newton’s second law. If the angle in formula 7 is zero, formulas 6 and 7 become identical.
5 Analysis

This chapter describes the scenario that was created for the simulation as well as analyse the results. The results are filtered as such that there is no irrelevant data due to the sheer amount of entries in the final result.

5.1 Result of the simulation

The simulation start by the focus vehicle accelerating up a 25 degree slope for two seconds, continue accelerating for one more second after which the vehicle continues to roll freely and only encountering rolling resistance and air resistance. This rolling continues until the fifth second where the vehicle will start steering right for two seconds. After the following two seconds, the vehicle continues rolling until the tenth second mark.

<table>
<thead>
<tr>
<th>Coord_x</th>
<th>Coord_y</th>
<th>Coord_z</th>
<th>Gyro_x</th>
<th>Gyro_y</th>
<th>Gyro_z</th>
<th>Accel_x</th>
<th>Accel_y</th>
<th>Accel_z</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.82E-06</td>
<td>1.78E-06</td>
<td>0</td>
<td>0.086323</td>
<td>0</td>
<td>0</td>
<td>9.8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.15E-05</td>
<td>5.35E-06</td>
<td>0</td>
<td>0.086323</td>
<td>0</td>
<td>0</td>
<td>7.077778</td>
<td>2.722222</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2.29E-05</td>
<td>1.07E-05</td>
<td>0</td>
<td>0.086323</td>
<td>0</td>
<td>0</td>
<td>7.077778</td>
<td>2.722222</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>3.82E-05</td>
<td>1.78E-05</td>
<td>0</td>
<td>0.086323</td>
<td>0</td>
<td>0</td>
<td>7.077778</td>
<td>2.722222</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>5.73E-05</td>
<td>2.67E-05</td>
<td>0</td>
<td>0.086323</td>
<td>0</td>
<td>0</td>
<td>7.077778</td>
<td>2.722222</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>8.03E-05</td>
<td>3.74E-05</td>
<td>0</td>
<td>0.086323</td>
<td>0</td>
<td>0</td>
<td>7.077778</td>
<td>2.722222</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>0.000107</td>
<td>4.99E-05</td>
<td>0</td>
<td>0.086323</td>
<td>0</td>
<td>0</td>
<td>7.077778</td>
<td>2.722222</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>0.000138</td>
<td>6.42E-05</td>
<td>0</td>
<td>0.086323</td>
<td>0</td>
<td>0</td>
<td>7.077778</td>
<td>2.722222</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>0.000172</td>
<td>8.02E-05</td>
<td>0</td>
<td>0.086323</td>
<td>0</td>
<td>0</td>
<td>7.077778</td>
<td>2.722222</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>0.00021</td>
<td>9.8E-05</td>
<td>0</td>
<td>0.086323</td>
<td>0</td>
<td>0</td>
<td>7.077778</td>
<td>2.722222</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>0.000252</td>
<td>0.000118</td>
<td>0</td>
<td>0.086323</td>
<td>0</td>
<td>0</td>
<td>7.077778</td>
<td>2.722222</td>
<td>0</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 1 Data from the first ten milliseconds of the simulation

The data shown in Table 1 displays the growth of x and y coordinates, as well as demonstrates the accelerometer reading understanding that a change of velocity has happened. The readings of Accel_x and Accel_y remain constant due to the fact that climbing up the slope, the rate of acceleration stays constant. The Gyro_x readings also indicate that the device is currently not parallel with the earth.
Table 2 shows the change of $Coord_z$ variable meaning the vehicle is now moving on the $z$ axis as well. The $Accel_x$ and $Accel_z$ readings show that the simulation recognizes the change in direction.

Table 2 Data from the vehicle starting a right turn

<table>
<thead>
<tr>
<th>Coord_x</th>
<th>Coord_y</th>
<th>Coord_z</th>
<th>Gyro_x</th>
<th>Gyro_y</th>
<th>Gyro_z</th>
<th>Accel_x</th>
<th>Accel_y</th>
<th>Accel_z</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>30.52131</td>
<td>3.566422</td>
<td>0</td>
<td>0.086323</td>
<td>0</td>
<td>0</td>
<td>9.8</td>
<td>0</td>
<td>0</td>
<td>4996</td>
</tr>
<tr>
<td>30.53037</td>
<td>3.566422</td>
<td>0</td>
<td>0.086323</td>
<td>0</td>
<td>0</td>
<td>9.8</td>
<td>0</td>
<td>0</td>
<td>4997</td>
</tr>
<tr>
<td>30.53943</td>
<td>3.566422</td>
<td>0</td>
<td>0.086323</td>
<td>0</td>
<td>0</td>
<td>9.8</td>
<td>0</td>
<td>0</td>
<td>4998</td>
</tr>
<tr>
<td>30.54848</td>
<td>3.566422</td>
<td>0</td>
<td>0.086323</td>
<td>0</td>
<td>0</td>
<td>9.8</td>
<td>0</td>
<td>0</td>
<td>4999</td>
</tr>
<tr>
<td>30.55754</td>
<td>3.566422</td>
<td>0</td>
<td>0.086323</td>
<td>0</td>
<td>0</td>
<td>9.8</td>
<td>0</td>
<td>0</td>
<td>5000</td>
</tr>
<tr>
<td>30.5666</td>
<td>3.566422</td>
<td>3.02E-06</td>
<td>0.086323</td>
<td>0</td>
<td>0</td>
<td>9.8</td>
<td>0</td>
<td>0</td>
<td>5001</td>
</tr>
<tr>
<td>30.57565</td>
<td>3.566422</td>
<td>9.06E-06</td>
<td>0.086323</td>
<td>0</td>
<td>0</td>
<td>9.793471</td>
<td>0</td>
<td>0.006529</td>
<td>5002</td>
</tr>
<tr>
<td>30.58471</td>
<td>3.566422</td>
<td>1.81E-05</td>
<td>0.086323</td>
<td>0</td>
<td>0</td>
<td>9.790209</td>
<td>0</td>
<td>0.009791</td>
<td>5003</td>
</tr>
<tr>
<td>30.59377</td>
<td>3.566422</td>
<td>3.02E-05</td>
<td>0.086323</td>
<td>0</td>
<td>0</td>
<td>9.78695</td>
<td>0</td>
<td>0.01305</td>
<td>5004</td>
</tr>
<tr>
<td>30.60282</td>
<td>3.566422</td>
<td>4.53E-05</td>
<td>0.086323</td>
<td>0</td>
<td>0</td>
<td>9.783693</td>
<td>0</td>
<td>0.016307</td>
<td>5005</td>
</tr>
<tr>
<td>30.61188</td>
<td>3.566422</td>
<td>6.34E-05</td>
<td>0.086323</td>
<td>0</td>
<td>0</td>
<td>9.780438</td>
<td>0</td>
<td>0.019562</td>
<td>5006</td>
</tr>
</tbody>
</table>
6 Conclusion and future research

The thesis described the creation of a simulation for a real-time tracking algorithm based on multi-sensor fusion. The state of the art section gave a short overview of alternative methods for creating a simulation of a similar background and at the same time describing them in the context of this thesis was not optimal.

A simulation of an ideal world was created, meaning that there is no background noise in neither the gyroscope nor the accelerometer readings. This allows for the testing of the real-time tracking algorithm in ideal conditions.

The proposed simulation can be improved upon by introducing dynamic challenges for the simulation. Currently the simulation follows a static path of incline climbing, driving straight and then making a right turn. The simulation could also be improved by creating a GUI for the introduction of the new dynamic challenge creation.
References


Appendix

I. Usage

Prerequisites:

- Node.js
- TypeScript
- NPM

The installation of the prerequisites can be checked with the following commands:

- node -version
- tsc -version
- npm -version

Downloading dependencies: npm i
Compiling the TypeScript code: npm run build
Creating the CSV file: npm run csv
II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, Aleksander Avastu,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

1.1 reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

1.2 make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright, of my thesis

Simulating control data for a real-time tracking algorithm
supervised by Artjom Lind.

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 24.04.2020